UNIVERSIDAD NACIONAL
DE MAR DEL PLATA

FACULTAD DE
INGENIERÍA

PROYECTO FINAL

"Sistema de Identificación de Unidades Móviles"

Autores: Fernando A. Otero

Gastón A. Menéndez

Directores: Ing. Miguel Revuelta

Ing. Julio C. Doumecq

Ingeniería Electrónica
Facultad de Ingeniería, UNMDP
Año 2004

RINFI

REPOSITORIO INSTITUCIONAL

FACULTAD DE INGENIERÍA | UNMDP

RINFI es desarrollado por la Biblioteca de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata.

Tiene como objetivo recopilar, organizar, gestionar, difundir y preservar documentos digitales en Ingeniería, Ciencia y Tecnología de Materiales y Ciencias Afines.

A través del Acceso Abierto, se pretende aumentar la visibilidad y el impacto de los resultados de la investigación, asumiendo las políticas y cumpliendo con los protocolos y estándares internacionales para la interoperabilidad entre repositorios

Esta obra está bajo una Licencia Creative Commons Atribución- NoComercial-CompartirIgual 4.0 Internacional.

UNIVERSIDAD NACIONAL
DE MAR DEL PLATA

FACULTAD DE
INGENIERÍA

PROYECTO FINAL

"Sistema de Identificación de Unidades Móviles"

Autores: Fernando A. Otero

Gastón A. Menéndez

Directores: Ing. Miguel Revuelta

Ing. Julio C. Doumecq

Ingeniería Electrónica
Facultad de Ingeniería, UNMDP
Año 2004

AGRADECIMIENTOS

Queremos agradecer:

A nuestros padres y nuestras familias por apoyarnos en los momentos más difíciles a lo largo de toda nuestra carrera.

A los directores del proyecto por ayudarnos y orientarnos en todo el proyecto, facilitándonos información, instrumental y dedicándonos tiempo y paciencia.

Al Laboratorio de Mediciones Electrónicas, por prestarnos el espacio físico para los ensayos en innumerables ocasiones y al Pañol, por facilitarnos los distintos instrumentos de medición utilizados.

A los ingenieros Hernán Sendra y Hernán Slavin por prestarnos su ayuda durante el proyecto.

A todos los profesores que conocimos en nuestras carreras, no sólo por los conocimientos brindados, sino también por inculcarnos nuevas maneras de pensar..

A todos los compañeros y amigos que hemos tenido durante la carrera, por compartir momentos a nivel académico y humano.

ÍNDICE DETALLADO

CAPÍTULO 6 MEDICIONES

CAPÍTULO 7 CONCLUSIONES

DIAGRAMA ESQUEMATICO

APÉNDICE A SOFTWARE DE DESARROLLO

APÉNDICE B MICROCONTROLADOR MC68HC908JK1/JK3

APÉNDICE C PROGRAMACIÓN

RESUMEN

RESUMEN

Se desarrolló un sistema de identificación de unidades móviles que consiste en el diseño, desarrollo e implementación de un equipamiento compuesto por plaquetas y módulos que van colocados en los vehículos y una consola decodificadora ubicada en la central. Estas plaquetas y módulos son los encargados de enviar a la central los códigos de identificación mediante la transmisión de tonos normalizados, estos códigos se componen de secuencias de tonos audibles en la banda de 900-2500 Hz. Se utilizó el vínculo de comunicación existente en los móviles ya sea, VHF o UHF.

Cada vehículo posee un único código que lo identifica. Toda vez que se pulse el pulsador de transmisión, el móvil será identificado en la consola de la central.

El sistema se ha basado en la programación de dos microcontroladores MC68HC908JK1/JK3 que son miembros de la familia 08 de Motorola, caracterizados por su bajo costo y alto desempeño. Dentro de las principales características de estos microcontroladores se destaca que son de memoria Flash (Borrables y programables eléctricamente) , cuentan con diez canales de conversión A/D y permiten ser programados In- Circuit, lo que significa que mediante el programa Winide permite verificar la programación realizada en el propio circuito.

El primero de ellos se encarga de reconocer dígitos programables, por ejemplo la licencia del taxi, y a partir de éstos transmitir los tonos normalizados según un determinado protocolo en cada oportunidad que el móvil hace uso de la radio.

El segundo microcontrolador, ubicado en el equipo de la central, debe detectar los tonos recibidos y mostrar la información original en un display. El display de carácter empleado es el WM-C0802M-1YLYb de Wintek de dos líneas de ocho caracteres cada una, de fácil conexionado con el microcontrolador. Para la detección se cuenta además con un circuito de entrada cuya tarea es convertir la señal recibida (senoidal + ruido) en una onda cuadrada de frecuencia idéntica a la transmitida.

El resultado final del ensayo del equipo construido fue muy satisfactorio. El sistema cumplió con todos los objetivos de prestación y funcionamiento propuestos en la etapa de diseño.

Por otro lado, la implementación real del diseño le sirvió a los autores para adquirir experiencia sobre problemas prácticos como por ejemplo los ruidos y otro fenómenos que se presentan en el proceso de decodificación de señales.

En relación a las posibles modificaciones y ampliaciones de las prestaciones del sistema, se podrían agregar más opciones tales como una entrada de micrófono en el transmisor para situaciones de emergencia, incluir mayor información en la comunicación como por ejemplo el barrio en el que se encuentra el móvil, destino y duración estimada del viaje, etc.
Finalmente, y con pocas modificaciones este sistema se podría adaptar para comunicaciones con una central de alarma.

CAPÍTULO 1

INTRODUCCIÓN

Capítulo 1

INTRODUCCIÓN

Existen en el mercado, equipos comerciales que brindan una prestación similar, pero no se hallan muy difundidos debido a su alto costo. Los sistemas desarrollados en el exterior son por la misma razón, poco viables en este momento económico del país.

El objetivo que se ha planteado es obtener un equipamiento con las siguientes características:

- Fácil Instalación y Manejo; la programación del código del móvil resulta muy sencilla de realizar para cualquier usuario
- Tamaño Reducido, para que pueda ser instalado junto con la radio de transmisión.
- Diseño Flexible, gracias a la cual posee grandes perspectivas de ampliación y/o modificación.
- Aprovechar al máximo las capacidades del microcontrolador usado, apelando al uso de la mayoría de los módulos que posee.
- Interfase Sencilla con el Display LCD: al usar un Display Inteligente, éste puede conectarse directamente a las salidas del Microcontrolador.
- Costo Reducido, debido a los pocos componentes electrónicos empleados en el diseño.
- Baja Sensibilidad al Ruido, para la implementación n la práctica y la inserción en el mercado
- Codificación Disponible para 10.000 móviles (ampliable sin necesidad de modificar el hardware), a diferencia del existente en el mercado nacional, implementado para 1000 vehículos.
- Cumplimiento con el Protocolo de Comunicación de CCIR Formato EIA

CAPÍTULO 2

ANTEPROYECTO

Capítulo 2

ANTEPROYECTO

Al iniciarse este proyecto aparecieron varias alternativas en cuanto a su diseño, su programación y los elementos circuitales utilizados. Hubo que realizar entonces un análisis de costos, confiabilidad y reproducibilidad para el sistema a desarrollar.

El primer punto a definir fue qué tipo de microcontrolador usar. Existen muchas posibilidades, pero las dos principales familias de microcontroladores usados habitualmente para este tipo de aplicación son la familia HC08 de microcontroladores Motorola y la familia PIC16FXX de Microchip.

Al realizar un análisis más detallado de las características de cada uno de estos microcontroladores se encontró con dos perfiles ligeramente distintos:

Ambas familias fueron diseñadas para aplicaciones de bajo costo y alto desempeño, y están disponibles en gran variedad de presentaciones. Su principal diferencia radica en que la familia HC08 posee un set de instrucciones más amplio que simplifica la tarea del programador, en tanto que la familia PIC16FXX posee un set reducido de instrucciones, lo que da mayor versatilidad pero mayor dificultad a la vez a su uso.

Con respecto al costo de ambos microcontroladores, no existen grandes diferencias, ni resulta significativo para este tipo de proyecto.

Los siguientes aspectos a considerar fueron relativos al programa a desarrollar:

- Generación de Frecuencias (Método de Generación, Tipo de Señal de Modulación). Las frecuencias pueden ser generadas por diferentes métodos, pero dadas las condiciones de confiabilidad y reproducibilidad es conveniente un diseño más general. Los esquemas de generación cíclicos no resultan particularmente confiables. Dadas las aplicaciones de las que dispone el micro usado, es posible recurrir a una de estas herramientas ya incorporadas, haciendo el diseño más fácil para el programador y más seguro para el usuario.

Los tonos transmitidos no tienen restricción alguna en cuanto a su forma (senoidal, triangular, cuadrada, etc.), de modo que es necesario tomar una decisión al respecto. Existen notas de aplicación para implementar un sistema que usa ondas de tipo senoidal (AN1771). Sin embargo, este tipo de onda trae complicaciones tanto en el software como en el hardware, aumentando costos y complejidad. Por esto se optó por modular con onda cuadrada, cuya generación es en forma directa, y si bien no es la recomendable para la modulación, resulta igualmente útil.

- Sistema de Detección de Frecuencias. Dado el perfil del proyecto es recomendable reducir el número de componentes electrónicos. Las características del microcontrolador utilizado nos posibilitan no solo comandar el display LCD sino también realizar las tareas de detección de frecuencias. Al tomar esta decisión, se ahorran varios dispositivos electrónicos (contadores electrónicos, flip-flops, etc.).

El tercer punto de discusión fue la forma de ingresar el código. La primera idea fue incluir dentro del programa, el código de cada móvil. La ventaja de este sistema es que no agrega costos adicionales y simplifica la programación, pero a los fines prácticos y comerciales del proyecto, esta opción no resulta viable ya que no es posible realizar modificación alguna por parte del usuario o técnico encargado de la instalación del sistema. Para cumplir este último requisito, el ingreso del código debe ser necesariamente por hardware y no por software. Para este propósito, fueron diseñados dos sistemas: uno implementado con un cuádruple dip-switch y otro usando un divisor resistivo a través de los conversores analógico-a-digital propios del micro. Estos sistemas pueden ser vistos con mayor detenimiento en el Cap.

CAPÍTULO 3

PROYECTO

Capítulo 3

PROYECTO

3.1 SISTEMA TRANSMISOR

3.1.1 OPCIONES DE PROGRAMACIÓN PARA LAS FRECUENCIAS NORMALIZADAS

Una de las piezas clave para el desarrollo del sistema de comunicación es el método de programación para obtener las frecuencias buscadas. Y para ello, hay que considerar varios elementos a tener en cuenta, tales como la exactitud obtenida en la generación de dichas frecuencias, la generalidad y flexibilidad del programa a la hora de modificar dichas frecuencias, la forma de onda deseada, complejidad en el diseño y optimización del código.

En el caso de este sistema, dado que se trata con varias frecuencias cercanas en el rango de audio, así como existe un margen de error de frecuencia bien determinado en el protocolo, una de las prioridades a tener en cuenta es, sin duda, la exactitud en la generación. Dicha exactitud está determinada tanto por el programa mismo como por la elección del cristal a usar.

El primer acercamiento posible es el de generar una onda cuadrada mediante ciclos de contadores que sucesivamente alternan “ceros” y “unos” lógicos a la salida de algún puerto. Este sistema es el más simple de imaginar, y es útil siempre que no se desee una forma de onda más allá de una cuadrada. Dicho programa tendría un esquema del tipo siguiente:

```
bclr / bset        i,PORTj
ciclo1     dbnz contador1,ciclo1
```

```
ciclo2      dbnz contador2,ciclo2
   .              .
   .              .
   .              .
ciclon      dbnz contadorN,ciclon
```

En donde, el tiempo T_0 en estado alto/bajo del bit i del port j está dado por el contenido de los registros contadores (($contador_i$)), la cantidad de ciclos o contadores (N) y el tiempo de ciclos de ejecución de la instrucción dbnz (tINS),
donde se incluye la frecuencia de operación del microcontrolador.

$T_0 = [(contador_1)+ (contador_2)+.....+ (contador_N)]* tINS$

En este caso, mientras se deba obtener frecuencias más bajas, esto es, períodos más altos, más alto debe ser el contenido de los contadores que para nuestro microcontrolador se ve limitado a 8 bits o sea hasta 2^8-1= 255 . Por lo tanto, con esta técnica para aumentar el período N veces, necesitamos N contadores o bien "recargar" el mismo N veces.

Otra posibilidad, un poco más intrincada, es usar una estructura del estilo de ciclos de repetición anidados, con lo que el tiempo en un estado alto o bajo, se torna proporcional al producto del contenido de los contadores.
El esquema para este caso sería:

```
bclr / bset     i,PORTj
ciclo1      dbnz contador1,ciclo1
ciclo2      dbnz contador2,ciclo1
   .              .
   .              .
   .              .
ciclon      dbnz contadorN,ciclo1
```

Y en este caso,

$T_0 = [(contador_1)* (contador_2)*.....* (contador_N)]* tINS$

Con lo cual, si bien se pueden lograr períodos mayores con mayor sencillez, la propagación de los errores resulta ser mucho mayor y esto limita su eficiencia.

El principal problema en estos planteos sencillos es que resultan -como se puede apreciar- poco prácticos para numerosas y bajas frecuencias. Para cada frecuencia, habría que calcular el contenido de cada contador para minimizar el error. Y a esto se le agrega el hecho de que mientras más ciclos haya (aún peor es la situación si éstos son anidados) más posibilidad de errores hay.

Por otro lado, se cuenta con varios módulos del microcontrolador que se han desarrollado para simplificar nuestra tarea a la hora de programar. El más importante para nuestros propósitos es el Módulo de Interfaz del Timer o TIM (Timer Interface Module).(Un análisis más exhaustivo puede verse en el Apéndice B -)

Los submódulos más importantes con los que cuenta el TIM, que se han utilizado para el diseño del sistema transmisor son Output Compare y Timer Overflow

OUTPUT COMPARE

La función Output Compare realiza básicamente una comparación entre un contador (TIM Counter) y un valor almacenado en un registro predefinido TIM Channel Register (TCH0). Dado el momento en que el contador alcanza el valor de este registro, la salida pasa a un estado determinado a priori en la configuración. De modo que si se configura adecuadamente el valor guardado en el TCH0, es posible variando los valores lógicos de la salida obtener una cuadrada de cualquier frecuencia deseada. Esta es la idea. Y ya que esta función cuenta con la posibilidad de que el valor lógico de la salida vaya alternándose cada vez que el contador alcanza ese valor determinado, se puede lograr con facilidad una onda cuadrada con un ciclo de trabajo del 50%, cargando al TCH0 con un número representativo al semiperíodo de la frecuencia a generar. Se puede observar el cálculo de estos valores en el Cap. –Protocolo Empleado. Este método reduce los errores en muchas formas: puesto que es idéntico el proceso para variar la salida de alto a bajo y viceversa, sólo es necesario optimizar uno de éstos. Asimismo, como ya se destacó, cualquier otro desarrollo basado en bucles y comparaciones conllevan mayores errores. Por último, está de manifiesto la sencillez de este método que reduce el hardware a cero y la programación a un mínimo posible.

TIMER OVERFLOW

La interrupción por desbordamiento del contador del Timer es fundamental por varias razones:

- Es el controlador de tiempos por excelencia. Sobre todo en un sistema como éste, en el cual el manejo de tiempos es de vital importancia.

- Es un valioso mecanismo para sincronizar las rutinas y las comunicaciones.

- Complementa a Output Compare para definir el formato de los tonos de transmisión.

Si bien con la función Output Compare se generan las cuadradas con las frecuencias deseadas, es necesario normalizarlas en el sentido de llevarlas al formato EIA del protocolo empleado. Para esto, aparece la interrupción por Timer Overflow que nos da la base de tiempo a utilizar. Debe controlar el tiempo de transmisión de señal (33 mseg) como también el tiempo sin tono (60mseg).

Para la medición de estos tiempos se usa el TIM Counter Modulo (TMOD), que genera una interrupción en el momento que el TIM Counter Register alcanza el valor almacenado en el primero.

Cabe destacar que mientras es mayor el uso de interrupciones del módulo TIM, surgen varias complicaciones en la programación. Esto se debe a que todas las funciones de este módulo usan un registro contador común (TIM Counter Register).

Una de las complicaciones que se pueden mencionar es la siguiente:

Puesto que se pretende obtener la mayor exactitud posible en las frecuencias generadas, es necesario usar el TIM Counter Register con la mayor resolución posible, esto es sin hacer uso de prescalers, con lo que la obtención de mayores períodos de $65536 / frec._{op}$ se hace más compleja.

También surgen problemas por el hecho de que el registro TCH0 usado por la función Output Compare debe variar su contenido cada vez que se alcance este valor. Esto sucede porque el TIM Counter Register es usado por varias aplicaciones de modo que no es posible realizar en él modificación alguna. Una vez que éste alcanza el valor del TCH0, continúa su conteo, y para poder realizar Output Compare nuevamente es necesario incrementar el valor cargado en TCH0. Por este motivo, ha sido necesario crear una rutina que vaya incrementando este contador con el valor calculado inicialmente. Es importante señalar que esto no trae aparejado errores adicionales ya que en tanto que esta rutina tarde en su ejecución menos que el semiperíodo del tono a transmitir, el programa funciona normalmente.

INTERRUPCIONES USADAS

Varias de las interrupciones usadas fueron vistas en la sección anterior. Tal es el caso de la función 'Output Compare' y 'Timer Overflow'.

En este sentido, Output Compare se usa adicionalmente para obtener el modo Emergencia. Lo que ocurre es que es necesario generar una interrupción externa por pulsador. Si bien es posible usar indistintamente cualquier entrada siempre resulta más conveniente una interrupción prearmada. En este caso, usando el canal 1 de la función OC se cumple este objetivo, sin necesidad de recurrir a la lectura cíclica de alguna entrada durante el lazo principal. Por último, a modo informativo debe decirse que OC no es una interrupción "tradicional": no debe usarse RTI para volver a la línea de programa original.

La transmisión en Modo Normal se ejecuta a través de la interrupción IRQ., que también posee una función adicional. Se encarga de cargar cada dígito del código de transmisión si se encuentra habilitada la opción de ingreso de código por dip-switch.

Finalmente, para el ingreso del código mediante los conversores A/D se ha usado la interrupción de conversión analógico-digital completa, cuya función resulta evidente: indica el fin de la conversión a realizar.

3.1.2 ELECCIÓN DEL CRISTAL – TOLERANCIA DE ERRORES

En principio, es lógico suponer que el cristal debe ser de la mayor frecuencia de resonancia posible, para así obtener la mayor resolución posible. Sin embargo, existen otros factores a considerar que demuestran que esto no es tan así. En primer lugar, es sabido que cualquiera sea el cristal, éste tiene un error de corrimiento asociado, cuyo rango viene dado generalmente en partes por millón. De modo que naturalmente, el cristal con mayor frecuencia lleva más posibilidad de corrimiento. En segundo término, el microcontrolador empleado maneja cristales con frecuencias limitadas. Por último, se debe ver la resolución y exactitud en verdad requerida. Es necesario entonces realizar un análisis más concreto.

Dada la fórmula para el cálculo de los valores para cargar el registro TCH0 en el transmisor (Cap. –Protocolo Empleado) se ve la dependencia directa del exactitud en el cristal en la exactitud obtenida en la frecuencia generada. Para cada número fijo correspondiente a todos los valores posibles podemos entonces traducir los máximos errores permitidos en las frecuencias generadas en los máximos errores de corrimiento en el cristal. Del cuadro 1 se obtiene que :

Frecuencia Cristal Mínima Tolerable: **9806336 Hz**.

Frecuencia Cristal Máxima Tolerable: **10199840 Hz.**

Error Relativo (sobre Frecuencia Nominal 10 MHz.): **±1,936 %**

Que corresponden a los valores calculados para los casos más restrictivos.

Código	Frec. Mín Detección (Hz)	Frec. Máx Detección (Hz)	Frec. Mín Cristal (Hz)	Frec. Máx. Cristal (Hz)
0	1941	2024	9798168	10217152
1	1100	1148	9785600	10212608
2	1171	1222	9780192	10206144
3	1249	1301	9792160	10199840
4	1328	1386	9774080	10200960
5	1417	1477	9794304	10209024
6	1509	1571	9802464	10205216
7	1605	1673	9796920	10211992
8	1712	1785	9806336	10224480
9	1822	1900	9795072	10214400

A	**2352**	**2454**	**9803136**	**10228272**
B	**911**	**950**	**9795072**	**10214400**
C	**2200**	**2291**	**9803200**	**10208696**
D	**971**	**1012**	**9795448**	**10209056**
E	**2064**	**2154**	**9791616**	**10218576**

Cuadro 1. Rangos de Errores Tolerables para los Tonos / Frecuencia del Cristal

Asimismo, puede calcularse el error introducido en la generación de las frecuencias centrales de los tonos en el transmisor. Este error surge debido al uso de contadores que permiten obtener sólo valores submúltiplos de la frecuencia de operación del bus ($f_{XTAL}/4$).Esto puede verse en el cuadro 2.

Tono Nro.	Valor Cargado	Frec. Central del Tono	Frec. Generada Teórica	Error Relativo
0	631	1981	1980,98257	$8{,}8.10^{-6}$
1	1112	1124	1124,10072	9.10^{-5}
2	1044	1197	1197,31801	$2{,}7.10^{-4}$
3	980	1275	1275,5102	4.10^{-4}
4	920	1358	1358,69565	$5{,}1.10^{-4}$
5	864	1446	1446,75926	$5{,}3.10^{-4}$
6	812	1540	1539,40887	$3{,}8.10^{-4}$
7	763	1639	1638,26999	$4{,}5.10^{-4}$
8	716	1747	1745,81006	$6{,}8.10^{-4}$
9	672	1860	1860,11905	$6{,}4.10^{-5}$
A	521	2401	2399,23225	$7{,}4.10^{-4}$
B	1344	930	930,059524	$6{,}4.10^{-5}$
C	557	2246	2244,16517	$8{,}2.10^{-4}$

D	1261	991	991,276764	$2,8.10^{-4}$	
E	593	2109	2107,9258	$5,1.10^{-4}$	

Cuadro 2. Tabla Errores en Frecuencias Centrales

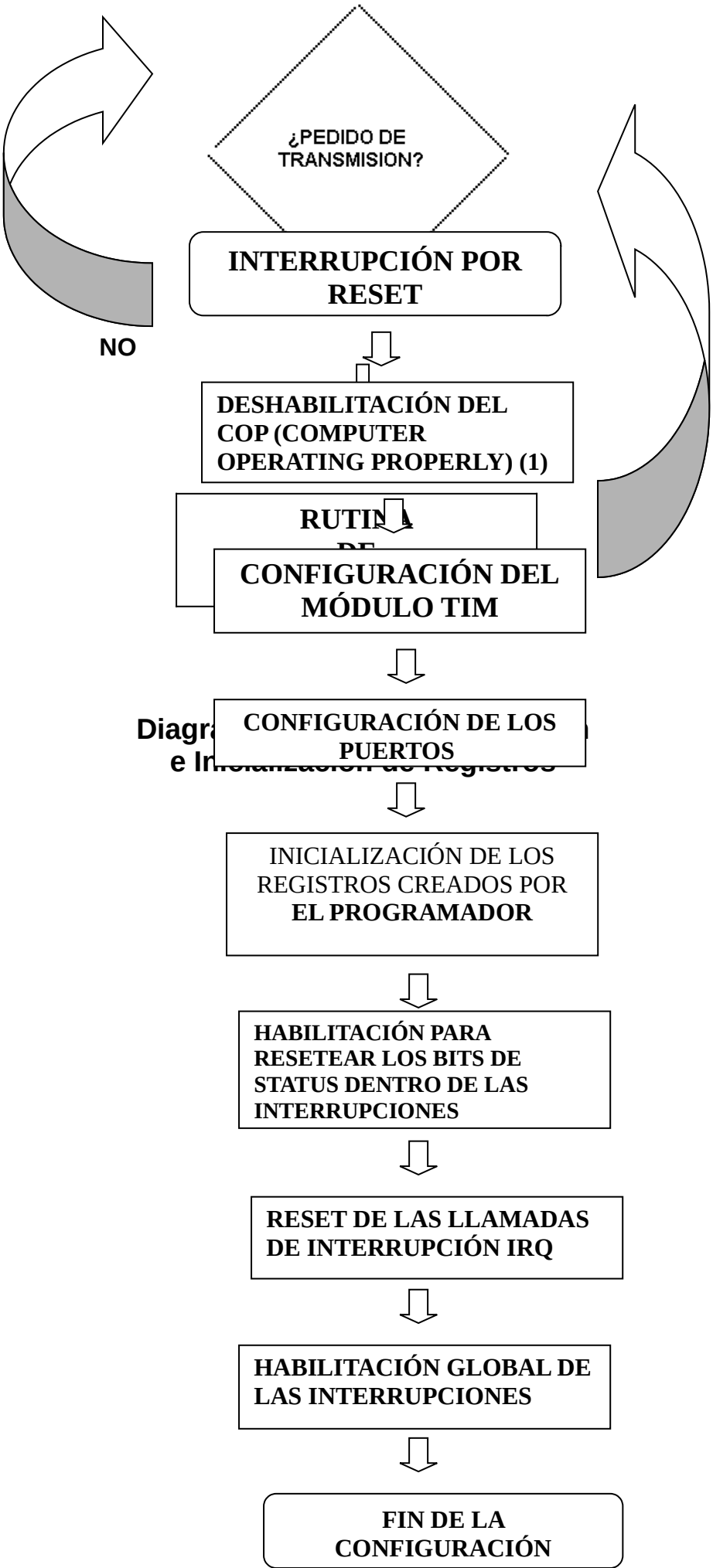
Nota: los cálculos realizados son válidos para un cristal de 10MHz. de frecuencia nominal.

3.1.3 TRANSMISOR - DIAGRAMAS DE FLUJO

Para el desarrollo del esquema y el diagrama de flujo del transmisor es necesario tener en cuenta cuáles son las funciones que éste debe cumplir. Si su única función es transmitir cuando se le solicita, el sistema se encuentra en estado inactivo la mayor parte del tiempo y lo más aconsejable es disminuir su complejidad en el programa principal y concentrarse en las rutinas de interrupción y el manejo de tiempos, en los que radica la mayor importancia.

Ahora se verán los diagramas de flujo del transmisor comenzando por los más generales hasta llegar a los más específicos, siguiendo las reglas del diseño modular:

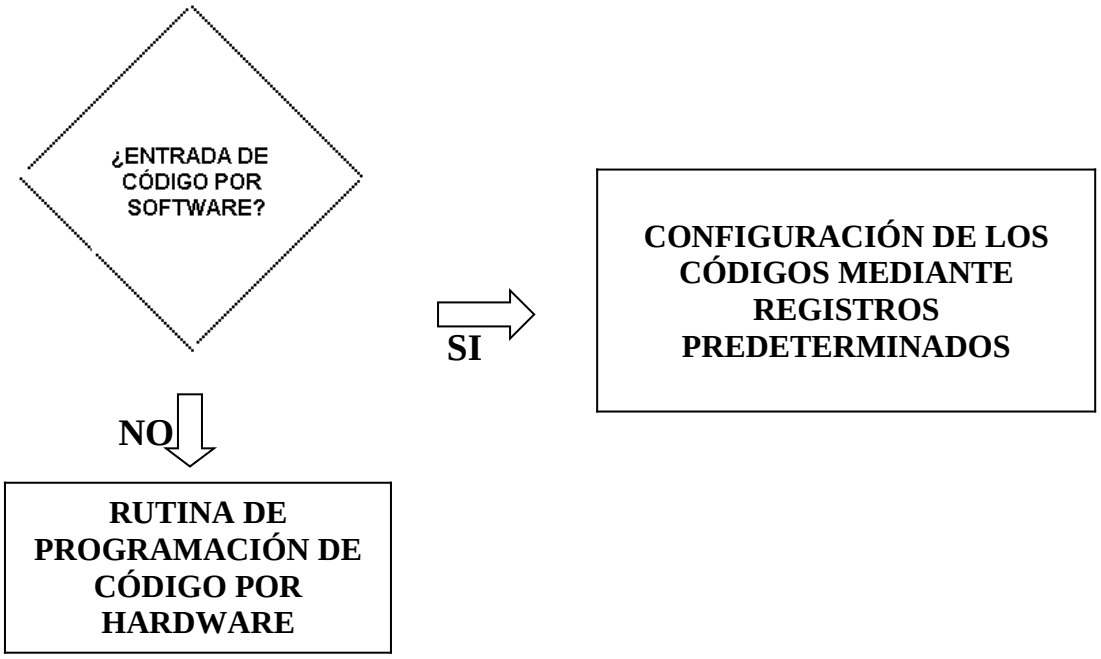
Diagrama de Flujo General


¿PEDIDO DE TRANSMISION?
NO
INTERRUPCIÓN POR RESET
DESHABILITACIÓN DEL COP (COMPUTER OPERATING PROPERLY) (1)
RUTINA DE
CONFIGURACIÓN DEL MÓDULO TIM
CONFIGURACIÓN DE LOS PUERTOS
INICIALIZACIÓN DE LOS REGISTROS CREADOS POR EL PROGRAMADOR
HABILITACIÓN PARA RESETEAR LOS BITS DE STATUS DENTRO DE LAS INTERRUPCIONES
RESET DE LAS LLAMADAS DE INTERRUPCIÓN IRQ
HABILITACIÓN GLOBAL DE LAS INTERRUPCIONES
FIN DE LA CONFIGURACIÓN

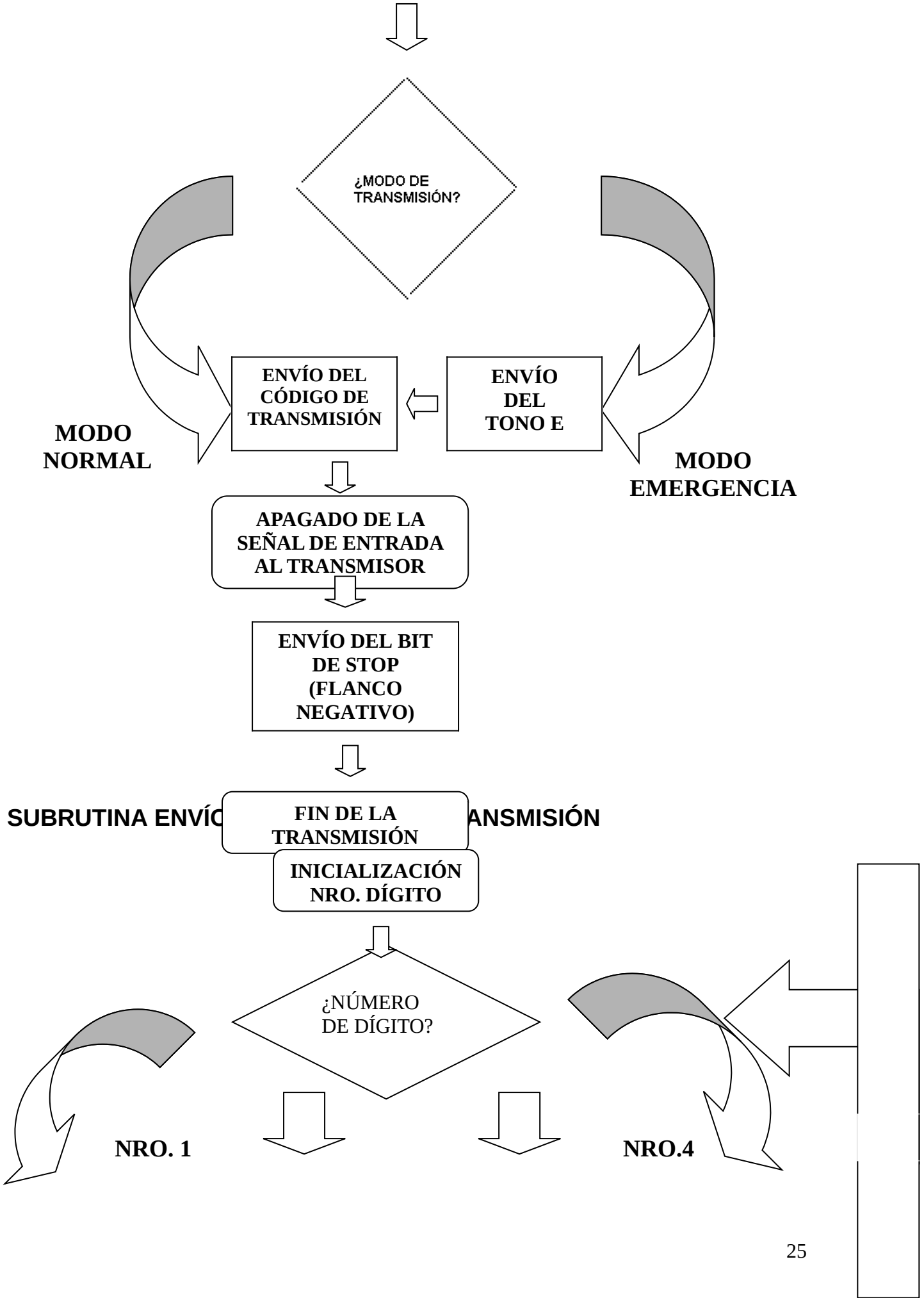
Diagra e In

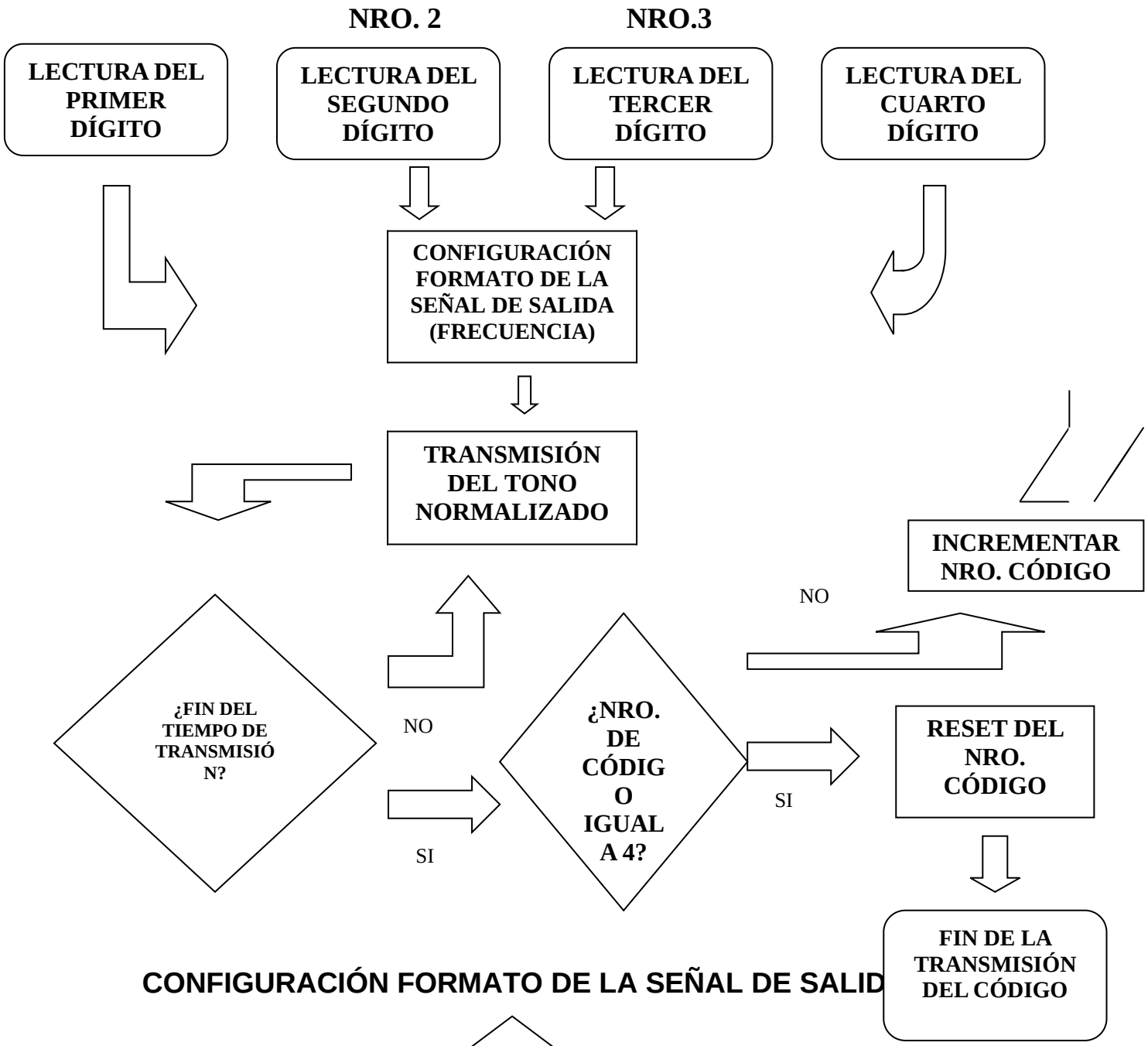
(1) El diagrama se puede desarrollar sin deshabilitar este registro. Sin embargo, las interrupciones que genera el COP a expensas de resolver posibles bucles o fallos en el funcionamiento del programa (ver Cap.), causarían demoras innecesarias y complicaciones en la programación.

PROGRAMACIÓN DEL CÓDIGO DE TRANSMISIÓN

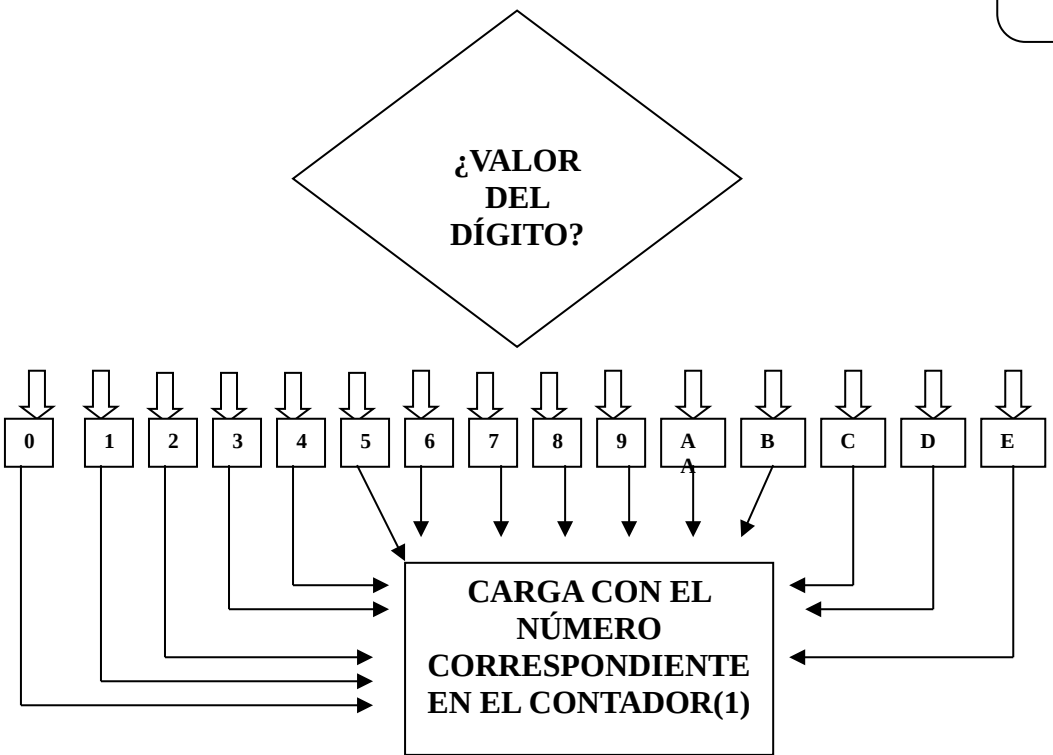


RUTINA DE TRANSMISIÓN


¿MODO DE TRANSMISIÓN?
ENVÍO DEL CÓDIGO DE TRANSMISIÓN
ENVÍO DEL TONO E
MODO NORMAL
MODO EMERGENCIA
APAGADO DE LA SEÑAL DE ENTRADA AL TRANSMISOR
ENVÍO DEL BIT DE STOP (FLANCO NEGATIVO)
SUBRUTINA ENVÍO
ANSMISIÓN
FIN DE LA TRANSMISIÓN
INICIALIZACIÓN NRO. DÍGITO
¿NÚMERO DE DÍGITO?
NRO. 1
NRO.4



CONFIGURACIÓN FORMATO DE LA SEÑAL DE SALID



CONFIGURACIÓN DEL RESTO DE LOS REGISTROS (2)

(1) El número que se almacena en el contador es función del cristal de operación del microcontrolador, la frecuencia deseada y el ciclo de trabajo de la señal. El contador puede ser creado por el usuario o bien uno de los preexistentes en el módulo del Timer. El cálculo para obtener dichos valores puede verse en el capítulo (Protocolo empleado).

(2) Dicha configuración depende del método de generación de frecuencia empleado y del microcontrolador empleado. Puede llegar a no ser necesaria. En el programa desarrollado para el sistema los registros a configurarse son TSC0 y TSC del módulo del Timer (Habilitación Contador del Timer e Inicialización de la función 'Output Compare')

3.1.4 INGRESO DEL CÓDIGO DE TRANSMISIÓN

Como ya se mencionó anteriormente, el código que se va a transmitir puede ser cargado desde memoria o bien, por hardware a través de los puertos de entrada del micro. En el diseño, para esta segunda opción enfrentamos varias posibilidades y ventajas y problemáticas respecto a cada una de ellas.

Dado que se trata de un código de cuatro dígitos de 0 a E en sistema hexadecimal, la lectura de cada uno de estos códigos implica el uso de cuatro pines de un puerto, por lo que, si se desea hacer una lectura de entradas en simultáneo, se necesitan 4x4=16 pines en total, a los que en realidad debería agregársele otro más para la lectura de la opción de origen de lectura del código. El microcontrolador empleado cuenta de por sí con 14 (con opción a 15) pines, de modo que dicha lectura no podría ser efectuada de manera directa.

De modo que es necesario idear un sistema de lectura o bien no simultáneo o bien por otro método. En nuestro caso, los parámetros principales considerados a la hora de decidirnos por un diseño son:

► Dimensiones expandidas a la plaqueta original del transmisor

► Complejidad del diseño

► Costo requerido

► Practicidad en su implementación

Y tras el análisis se llegó a dos posibles alternativas:

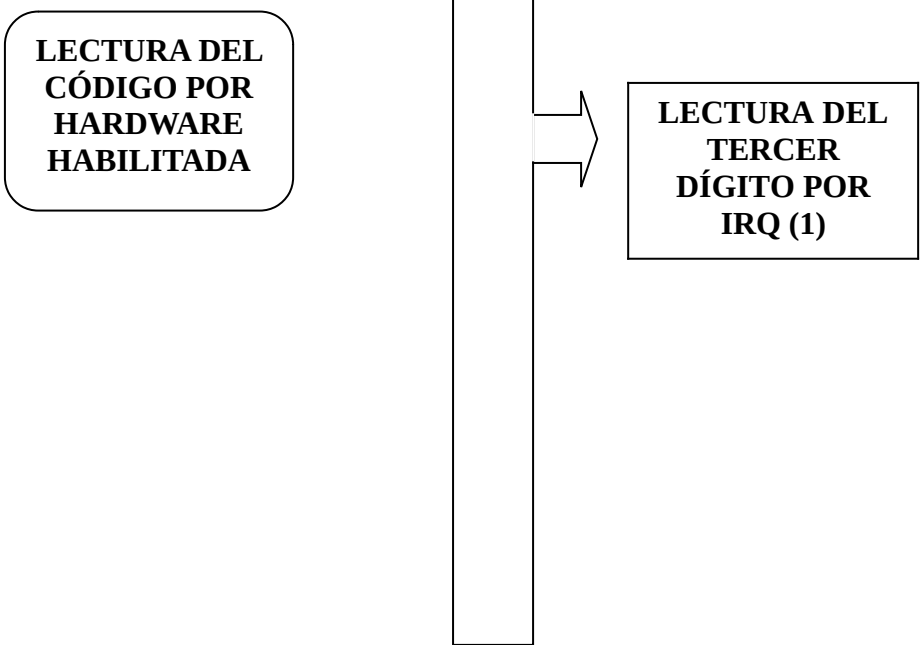
► Una lectura de cuatro entradas digitales que por pulsador van cargando código por código.

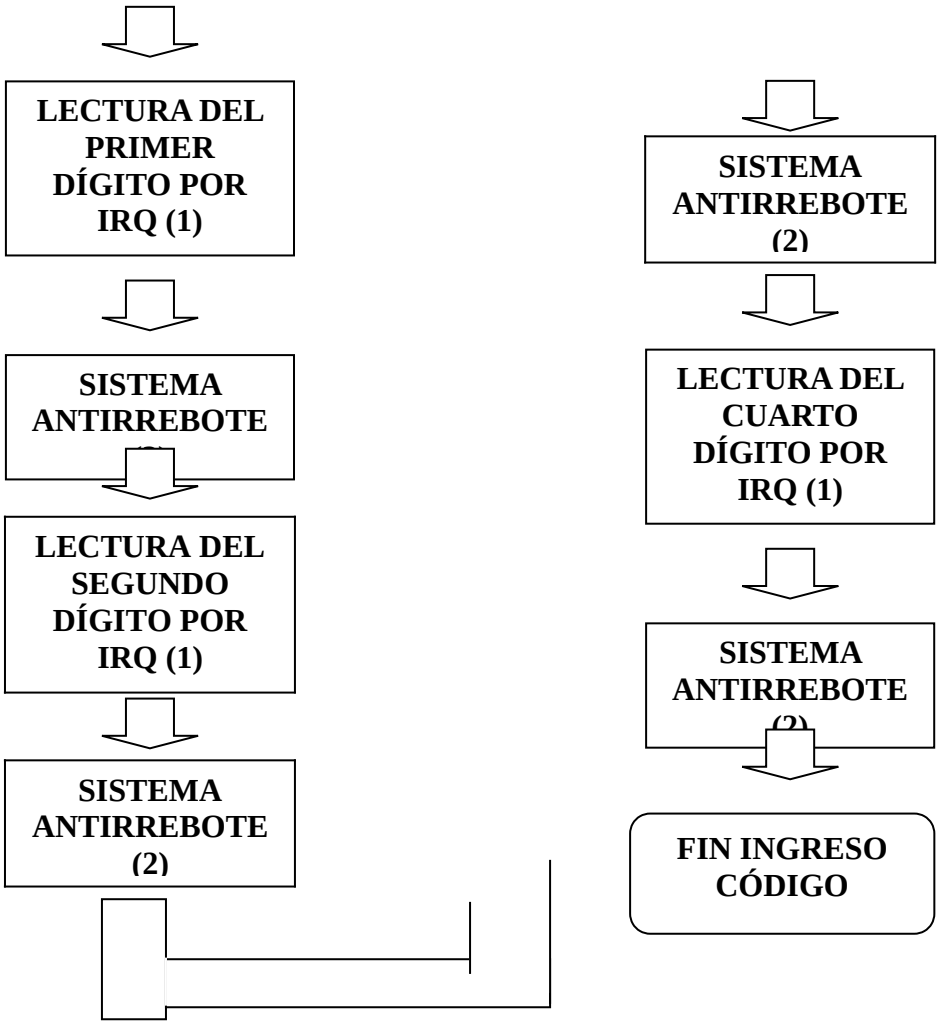
► Un sistema analógico con cuatro salidas hacia los canales de conversión A/D del microcontrolador, a modo de multiplexar cada código por cada canal.

Resulta entonces que el primer diseño es de mucha menor complejidad, tanto en el hardware como en el software (donde la mayor complejidad recae en el antirrebote hecho por soft para el pulsador), así como de tamaño más reducido. Por otro lado, la segunda alternativa presenta una gran ventaja que es la le imultánea. En este sentido, cabe señalar el hecho de que una lectura simultánea de todos l igos tiene notables mejoras en la implementación práctica del sistema ya que ello posib a lectura automática del código con la configuración correspondiente de la conversión A/D e obviamente facilita el manejo del usuario. En este caso, el análisis de costo no es de may evancia.

INGRESO DEL CÓDIGO M NTE DIP-SWITCH

El funcionamiento del sistem práctica puede verse en el Cap. – Manual de Operación. Se verá entonces el diagra flujo y su explicación correspondiente:





(1) La lectura del dígito se realiza tras la configuración previa del cuádruple dip-switch. Luego de esto, se produce la interrupción IRQ tras oprimir el pulsador de entrada de código EC/TMN.

(2) El sistema antirrebote tiene como función evitar que un solo pulsado sea tomado como si fueran varios, con lo que el programa funcionaría erróneamente. Esto se debe a que al apretar el pulsador, aparecen oscilaciones intermitentes (rebotes) que generan múltiples interrupciones. Para ello, se inhiben tanto las interrupciones como sus respectivos flags durante un período característico de entre 200 y 300 mseg.

ANÁLISIS DE LOS CONVERSORES A/D

INTRODUCCIÓN

Las características del módulo de ADC del MC68HC908JK1/JK3 incluyen:

• 12 Canales con Entradas Multiplexadas
• Aproximación Lineal Sucesiva con Monotonicidad
• Resolución de 8-bit
• Conversión Ünica o Continua
• Flag de Conversión Completa o Interrupción de Conversión Completa
• Clock ADC Seleccionable

• Tiempo de Conversión Mínimo de 16 µseg.

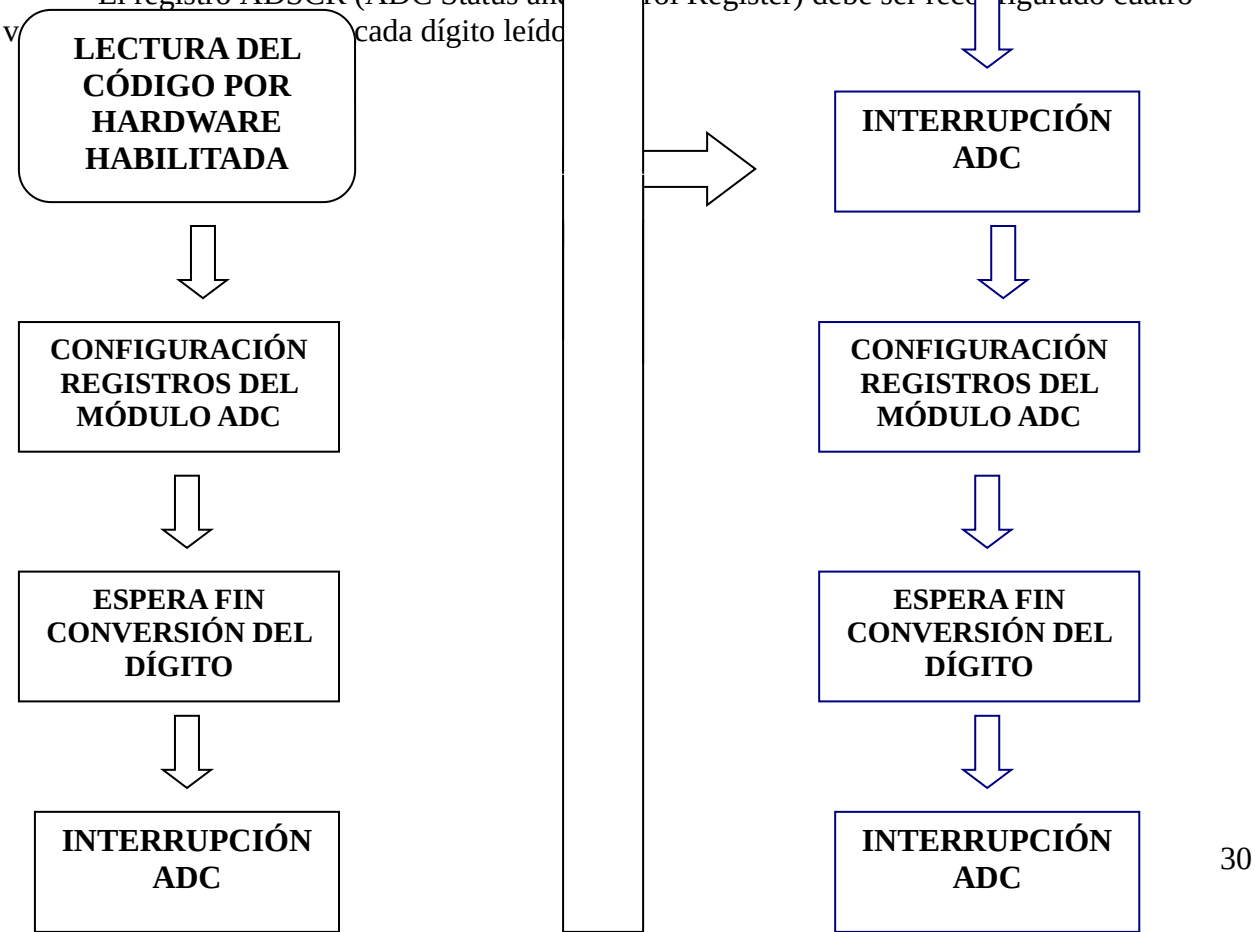
CONFIGURACIÓN

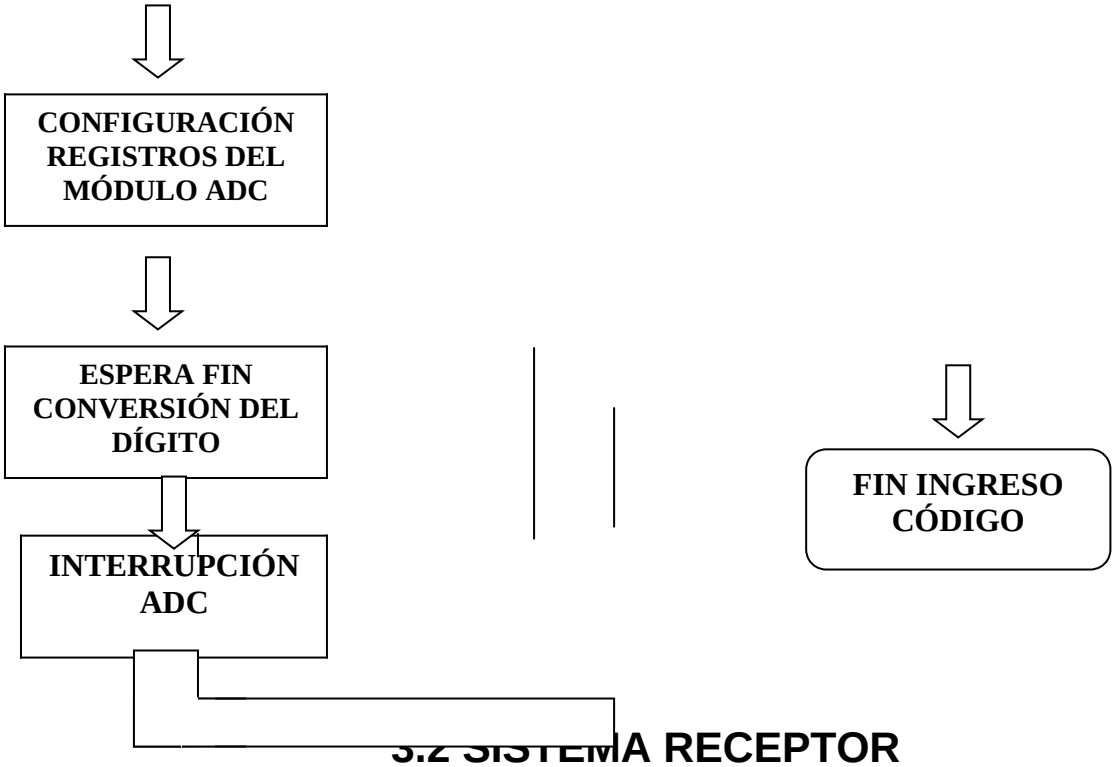
Todas las posibilidades vistas en el análisis del módulo ADC quedan abiertas para su uso. Dado que para nuestros fines no es necesario una conversión continua (sólo se sensan tensiones fijas), se ha optado por la conversión única que hace así despreciable el tiempo de conversión, en tanto que el uso del flag o la interrupción es prácticamente indistinto. Analicemos ahora la configuración de los registros pertinentes (fig. 1)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$003C	ADC Status and Control Register (ADSCR)	Read:	COC	EN	ADCO	CH4	CH3	CH2	CH1	CH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$003D	ADC Data Register (ADR)	Read:	AD7	D6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:								
$003E	ADC Input Clock Register (ADICLK)	Read:	ADIV	IV1	ADIV0				0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Figura 1. Registros de Entrada/Sa el Módulo ADC

El registro ADSCR (ADC Status an rol Register) debe ser reconfigurado cuatro v cada dígito leído





3.2 SISTEMA RECEPTOR

3.2.1 SISTEMA DE DETECCIÓN DE FRECUENCIA

El diseño probablemente más importante y de mayor complejidad en el sistema es el de detección de los códigos en el receptor porque éste comprende tanto el reconocimiento de las frecuencias transmitidas como el proceso de decisión final del código enviado.

Dada la normativa del protocolo usado, por cada código que se transmite se pueden recibir aproximadamente entre 30 y 85 períodos de frecuencia que desde ya no tienen por qué ser exactamente iguales entre sí. Con lo cual, no sólo es necesario una buena detección de cada frecuencia sino asimismo una buena correlación entre los valores obtenidos de modo de decidir correctamente el código enviado.

FUNCIÓN INPUT CAPTURE

Como se vio en el anteproyecto, se deseaba usar el propio microcontrolador como sistema detector de los tonos transmitidos. Para esto era necesario recurrir a otra de las funciones prediseñadas del HC908JK3: la función 'Input Capture'.

Puede verse una descripción un poco más detallada de su funcionamiento en el Cap. , pero básicamente lo podemos definir como un contador entre flancos. Se encarga de contar los ciclos del registro contador del Timer entre flancos programables, ya sean descendentes o ascendentes que llegan a una determinada entrada (PTB4/PTB5). Si se logra que a tal entrada llegue una señal cuadrada o similar, . de frecuencia igual a la transmitida con flancos de 0 a 5v, es posible calcular dicha frecuencia. De modo que realizando una conversión de los rangos de frecuencias para la decodificación hacia dichos valores, es posible lograr una buena detección. Se puede entonces, tabla mediante, obtener el dígito recibido. El cálculo del valor correspondiente a cada tono puede verse en el Cap. – Protocolo Empleado.

Entre las opciones a considerar está la polaridad del flanco a tener en cuenta. Si bien es posible una detección indiscriminada de flancos (tanto los positivos como los negativos), esto no es particularmente útil, salvo que se trate de señales con un 50% de ciclo de trabajo (idealmente el tipo de onda del transmisor). Es cierto que este tipo de medición posee la ventaja de duplicar el número de mediciones sobre cualquiera de las otras dos posibilidades. Sin embargo, hay que tener presente que el ciclo de trabajo de la señal detectada no es necesariamente el mismo que el de la transmitida, debido principalmente al rango de histéresis del comparador que es en verdad el que define dicha característica en la señal a la entrada del sistema detector.

Por otro lado, el uso de flancos negativos o positivos es prácticamente equivalente, salvo por la primera medición. En este caso, dado que el sistema inicia el proceso de detección con el flanco positivo, si es programado para la detección de flancos negativos, la primera medición será de aproximadamente la mitad del valor real u otro porcentaje dependiendo del ciclo de trabajo de la señal de entrada. Esto es debido al hecho de que el sistema de detección de flancos es activado automáticamente tras recibir el bit de start. Es posible entonces, con una pequeña pérdida de tiempo en la detección, corregir este error, aunque en realidad dicho error resulta despreciable en la práctica.

Una vez que se obtiene el número proporcional al período del tono recibido, es necesario realizar un análisis de "pertenencia" de dicho valor. Es decir, verificar si pertenece al conjunto de valores posibles y de ser así a cuál de todos ellos. Esta parte del método de detección puede parecer la más simple, pero no lo es tanto. Desde ya que resulta muy ineficiente y engorroso para la programación, comparar el valor hallado con cada uno de los posibles. Por otro lado, no hay muchas más alternativas. La idea básica es siempre la misma: comparar el número con los diferentes rangos posibles. Y aquí aparecen varias posibilidades. En realidad, no existe una opción óptima muy superior al resto. En este sentido, se sigue un método propio que consta de dos partes:

Análisis del Byte más Significativo : Dado que el Registro donde se guarda el número asociado al período de la señal de entrada (TCH0) es de 16 Bits, es necesario efectuar una doble comparación. Esta comparación tiene la particularidad de hacerse en sistema hexadecimal por simplificar el desarrollo (la división en dos bytes tiene un desarrollo muy sencillo en sistema hexadecimal). Dado que los posibles valores del MSB (Byte más Significativo) son relativamente pocos (un total de ocho) se efectúa una serie de comparaciones deL número obtenido y si no hay coincidencia alguna, se descarta. De otro modo, pasa al segundo análisis.

Análisis del Byte Menos Significativo: Aquí si es necesaria una comparación entre distintos umbrales. Según el valor determinado del primer byte, pasa a una determinada serie de comparaciones, cuyo orden varía en pro de conseguir el programa de menor código. Estas

comparaciones son necesariamente del tipo ‘mayor que’ o ‘menor que’ puesto que son las que en verdad definen los distintos rangos.

RUTINA DE DECISIÓN DEL DÍGITO

Con la detección de cada tono empleando ‘Input Capture’, queda ver entonces cómo decidir de qué dígito se trata. Porque un sistema de detección de frecuencia no es suficiente. Es necesario saber si los distintos tonos detectados tienen relación entre sí –como debiera ser- o no son más que producto de una comunicación fallida o una interferencia.

Por lo anterior, se necesita realizar un sistema de detección de correlación entre los distintos tonos por dígito. Existen varias posibilidades:

- Considerar un valor umbral de tonos en el mismo rango. Es decir, para n veces que se detectaron tonos correspondientes al mismo dígito, tomar dicho dígito como el recibido.

- Como variante del anterior, el umbral puede ser para detecciones consecutivas.

- Considerar el dígito en cuyo rango de frecuencias se detectaron más tonos.

Finalmente se optó por la tercer alternativa por resultar más sistemática. Se implementó con contadores para cada valor posible y un algoritmo de comparación. Además, dado que la cantidad de tonos recibidos depende de la frecuencia de éstos, un umbral fijo resultaría poco recomendable para una decisión correcta.

INTERRUPCIONES UTILIZADAS

Se hace uso, al igual que en el sistema transmisor, de la interrupción por Timer Overflow, con el mismo fin: como referencia para la medición de los tiempos.

Otra interrupción empleada es IRQ, que le permite al usuario, realizar un “congelado” de display.

Existen en el sistema receptor muchas cuestiones a discernir

MANEJO DEL DISPLAY LCD

Una de los aspectos que tiene considerable complejidad en la programación es el manejo del display LCD por parte del microcontrolador receptor. Lo que ocurre es que existen muchas cuestiones a definir en la configuración que se pueden agrupar en dos grandes clases:

- Inicialización
- Opciones de Visualización

Dentro de la Inicialización, cuyas especificaciones se detallan en el Cap. – Display LCD, aparecen: el momento de la inicialización, el manejo de los retardos, los tiempos de crecimiento y caída de los flancos, el orden conveniente en las instrucciones, etc.

Dentro del segundo grupo, las opciones de visualización más destacadas son el método de actualización del display, la visualización en tiempo real/ no real, los mensajes a definir y su formato, etc.

El análisis de la inicialización presenta un problema básico:

¿Cuándo se debe realizar?

No existe en sí un mejor momento para realizar la inicialización, si bien es preferible hacerlo en simultáneo durante el tiempo de la primera detección o bien apenas termine la configuración inicial del receptor. Esto es, si se desean aprovechar al máximo los tiempos. Lo que ocurre es que lleva un tiempo durante el cual el sistema está dedicado a esa sola tarea y si bien es posible hacerlo durante los tiempos de inactividad en la recepción, no sería lo aconsejable por posibles fallas que puedan ocasionar problemas en la comunicación. Es por esto que se considera que las mejores opciones son efectuar la inicialización fuera del tiempo de comunicación o luego de finalizar la primera detección. En el primer caso, posibles problemas relativos a la inicialización del display son puestos en evidencia inmediatamente. Si se ejecuta tras terminar la detección del primer código recibido, se tiene así como ventaja ejecutar toda la rutina de display en forma concatenada, como es sugerido. Finalmente se concluyó que lo más apropiado era una inicialización inmediata tras la configuración inicial del receptor, donde se incluyó además una rutina de presentación.

El manejo de los retardos repite el mismo problema de la generación de frecuencias en el transmisor:

¿Cómo conviene implementarlos?

En este sentido, puede hacerse uso del módulo del Timer, tal como se vio en el Cap - Opciones de Programación de Frecuencias Normalizadas, o bien un ciclo generador de retardos que, dado que no se trata de obtener exactitud crítica, resulta igualmente aceptable.

El tercer punto a definir es verificar que tiempos de crecimiento, caída, establecimiento y mantenimiento soporta el display. Para esto es necesario –tras un análisis del manual del fabricante- un ensayo de naturaleza meramente práctica, hasta lograr una inicialización exitosa

Las opciones de visualización si bien no son de una magnitud tan crítica como la inicialización del display, revisten también cierta importancia. Si se efectuara la inicialización tras la configuración inicial del receptor, se podría obtener una visualización casi inmediata para

cada dígito detectado y enviar por ejemplo, un mensaje de emergencia con mayor rapidez, evitando pérdidas de tiempo. Pero también es posible que comience mostrando un código que termine siendo un mensaje de error, generando confusión en su lectura Esta es entonces una de la cuestiones a determinar: si el display del código de transmisión será a tiempo real o no. Si se usa una inicialización

La actualización del display puede desarrollarse de diversas maneras: por ejemplo scrolling (mensaje que va desplazándose en forma cíclica a lo largo del display), forma alternada, sobreescritura, etc. La decisión de qué método elegir es básicamente indistinta, variando únicamente su presentación y complejidad en la programación.

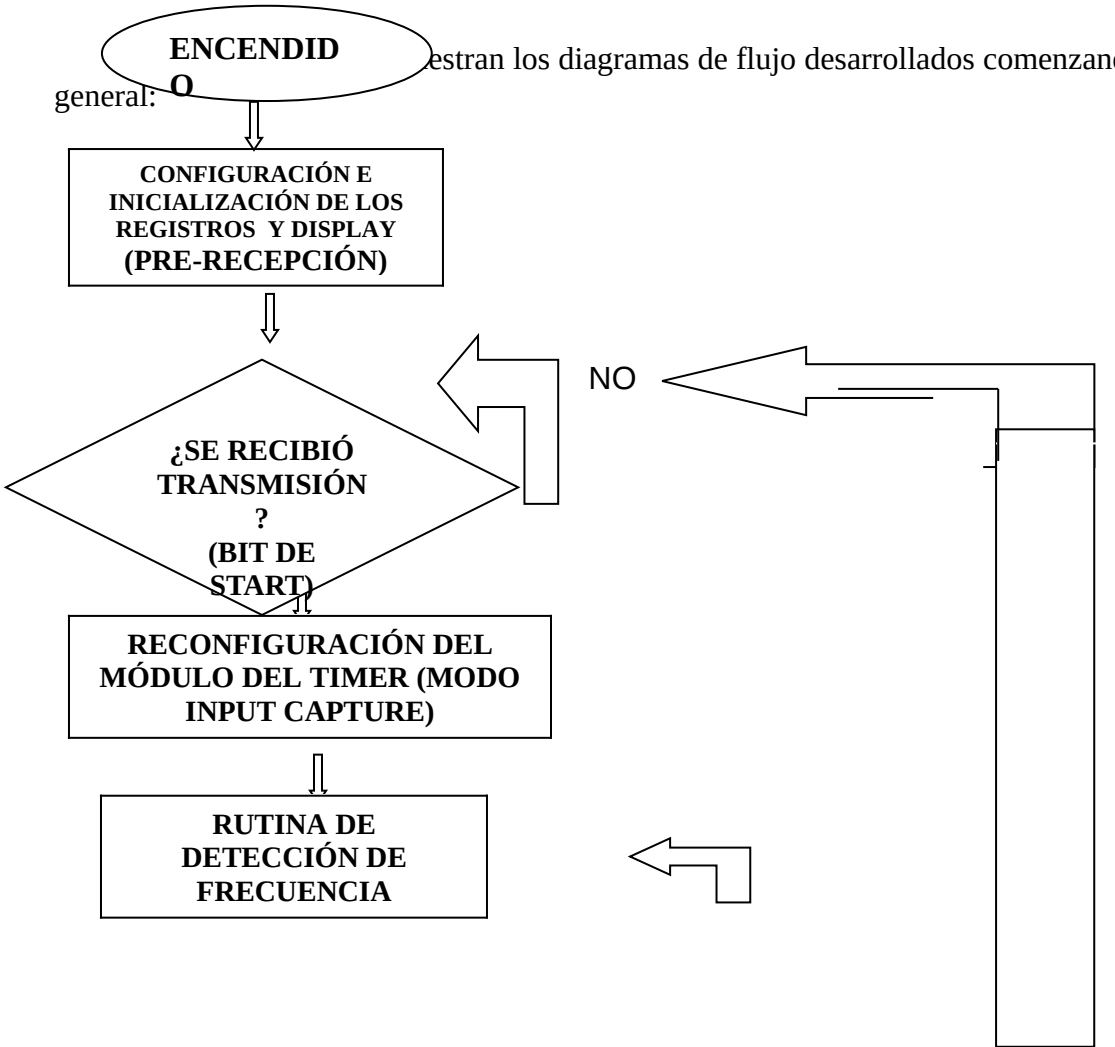
Finalmente, otro punto a definir es el formato del mensaje a mostrar. Es recomendable en este aspecto, para simplificar la programación, usar caracteres generales standard y emplear tablas de conversión a caracteres tipo ascii, que son los usados por el display LCD.

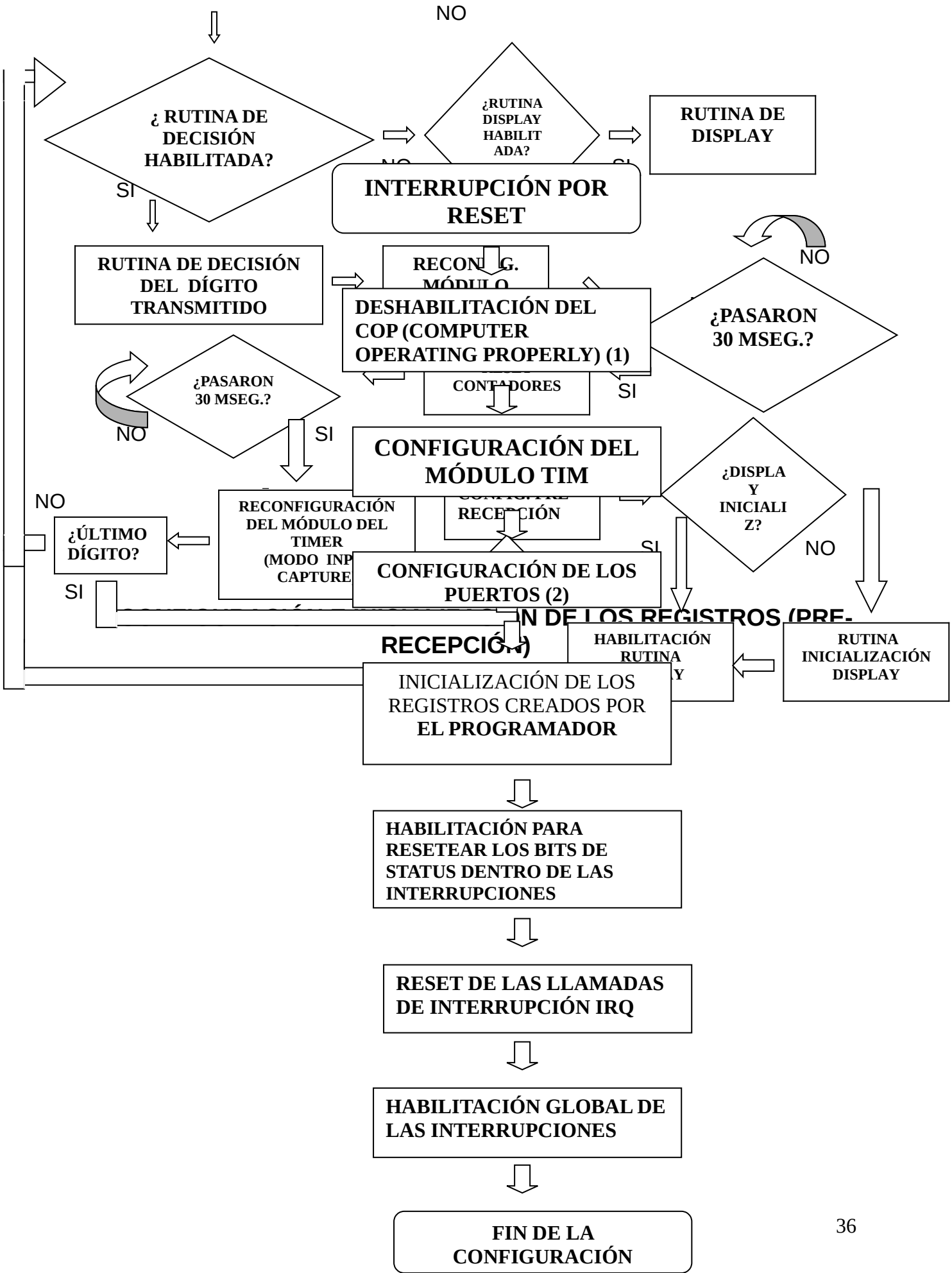
MODO FREEZING

El sistema receptor cuenta además con la posibilidad de realizar un congelado de display para casos tales como una transmisión en modo emergencia durante un intervalo de tiempo congestionado de comunicaciones. Para no perder las posibles transmisiones recibidas mientras se encuentra en este modo. En el diseño de esta opción aparecieron varias consideraciones sobre este modo. La principal cuestión era decidir qué dirección debía seguirse si se excedía la capacidad del buffer de almacenamiento. O bien se perdían transmisiones o bien el modo freezing debía ser abortado. Se optó por la segunda posibilidad, para mantener la máxima fiabilidad posible para el sistema.

3.2.2 DIAGRAMAS DE FLUJO DEL SISTEMA RECEPTOR

estran los diagramas de flujo desarrollados comenzando por el más general:

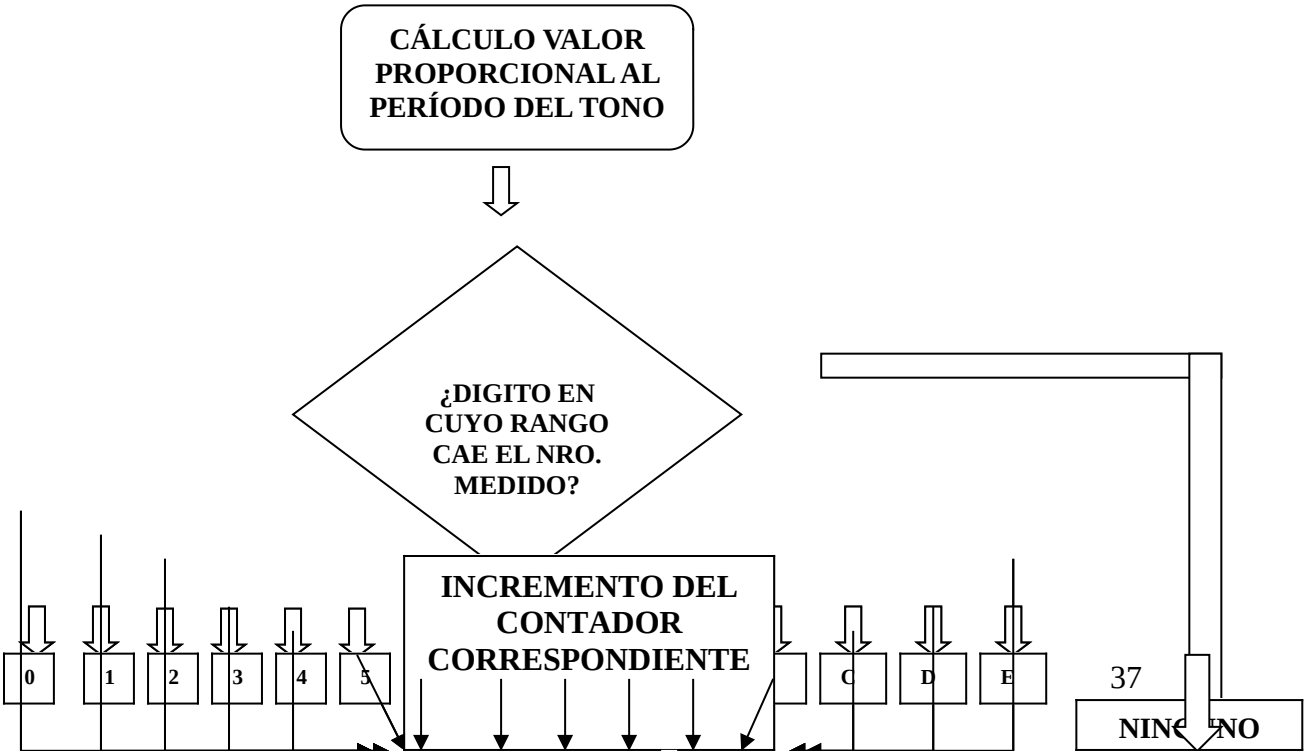


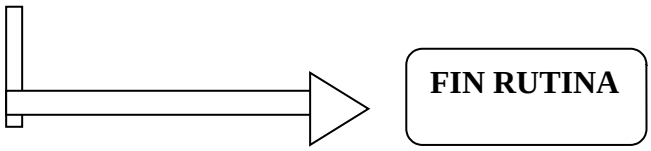

NO
¿ RUTINA DE DECISIÓN HABILITADA?
¿RUTINA DISPLAY HABILITADA?
RUTINA DE DISPLAY
SI
INTERRUPCIÓN POR RESET
RUTINA DE DECISIÓN DEL DÍGITO TRANSMITIDO
DESHABILITACIÓN DEL COP (COMPUTER OPERATING PROPERLY) (1)
¿PASARON 30 MSEG.?
NO
SI
CONFIGURACIÓN DEL MÓDULO TIM
¿DISPLAY INICIALIZ?
RECONFIGURACIÓN DEL MÓDULO DEL TIMER (MODO INP CAPTURE
RECEPCIÓN
¿ÚLTIMO DÍGITO?
CONFIGURACIÓN DE LOS PUERTOS (2)
RECEPCIÓN)
HABILITACIÓN RUTINA
RUTINA INICIALIZACIÓN DISPLAY
INICIALIZACIÓN DE LOS REGISTROS CREADOS POR EL PROGRAMADOR
HABILITACIÓN PARA RESETEAR LOS BITS DE STATUS DENTRO DE LAS INTERRUPCIONES
RESET DE LAS LLAMADAS DE INTERRUPCIÓN IRQ
HABILITACIÓN GLOBAL DE LAS INTERRUPCIONES
FIN DE LA CONFIGURACIÓN

(1) El diagrama se puede desarrollar sin deshabilitar este registro. Sin embargo, las interrupciones que genera el COP a expensas de resolver posibles bucles o fallos en el funcionamiento del programa (ver Apéndice B, pág 110), causarían demoras innecesarias y complicaciones en la programación.

(2) En realidad, los puertos son configurados todos, salvo el de encendido, como entradas.

RUTINA DETECCIÓN DE FRECUENCIA

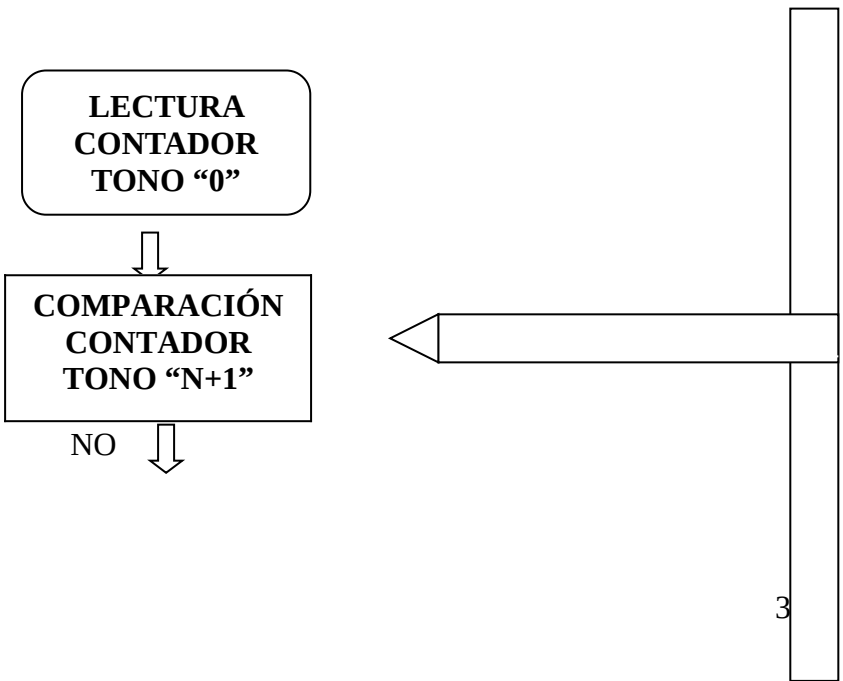


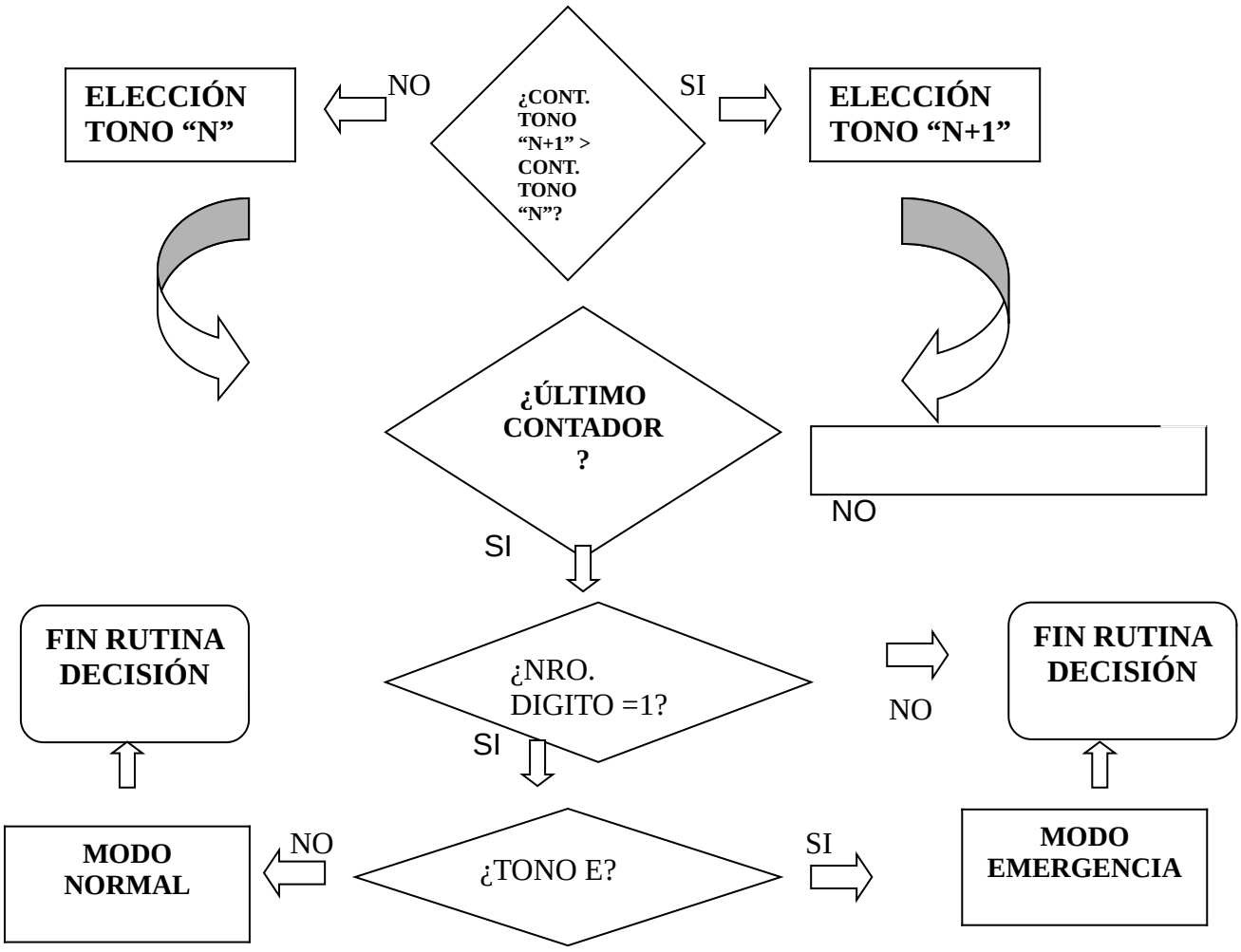


RUTINA INICIALIZACIÓN DISPLAY

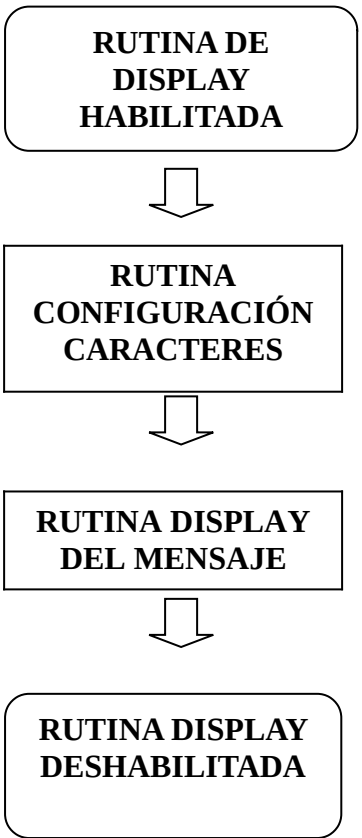
Ver Sección 3.4 –Display LCD

RUTINA DECISIÓN DE DÍGITO TRANSMITIDO

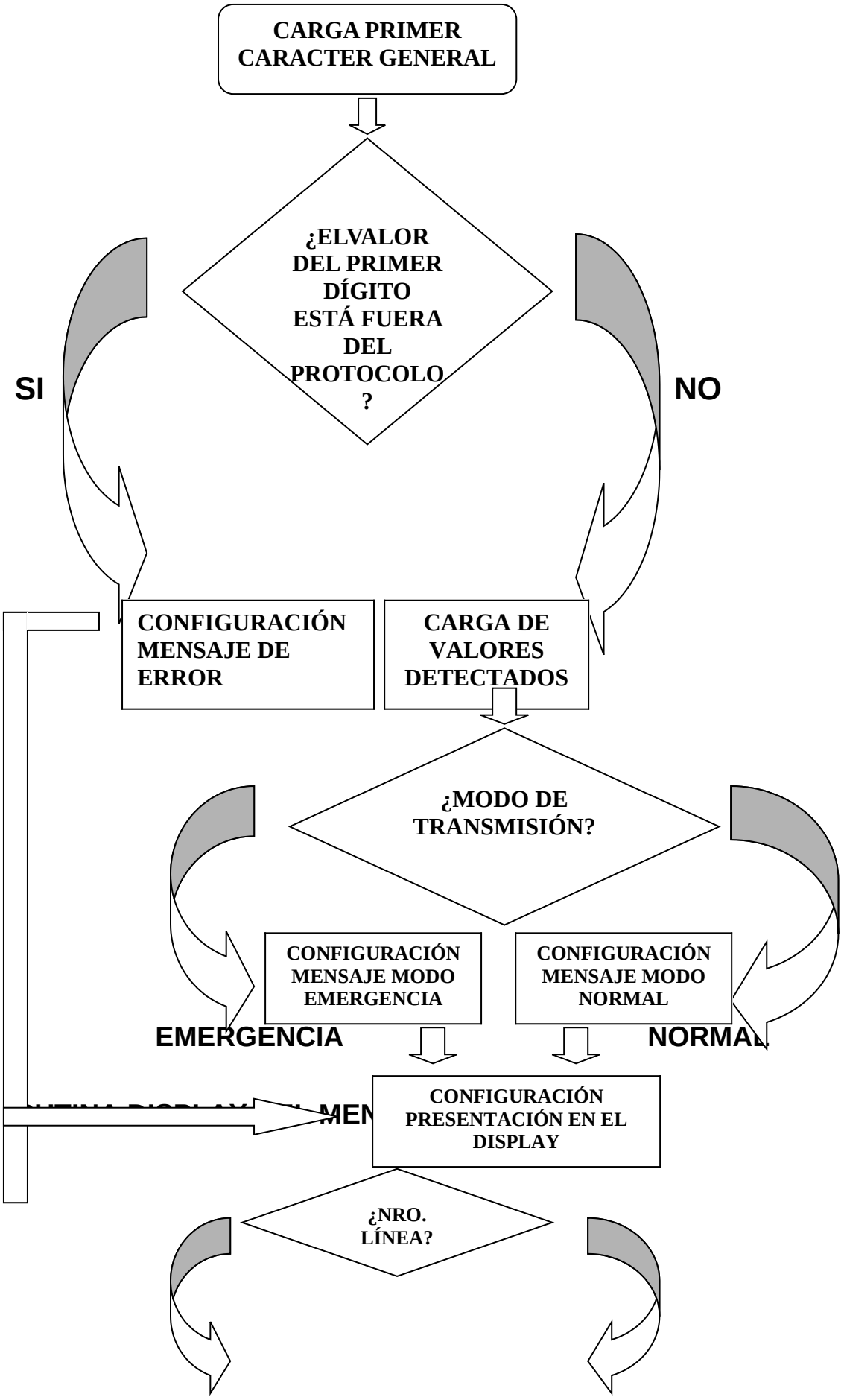


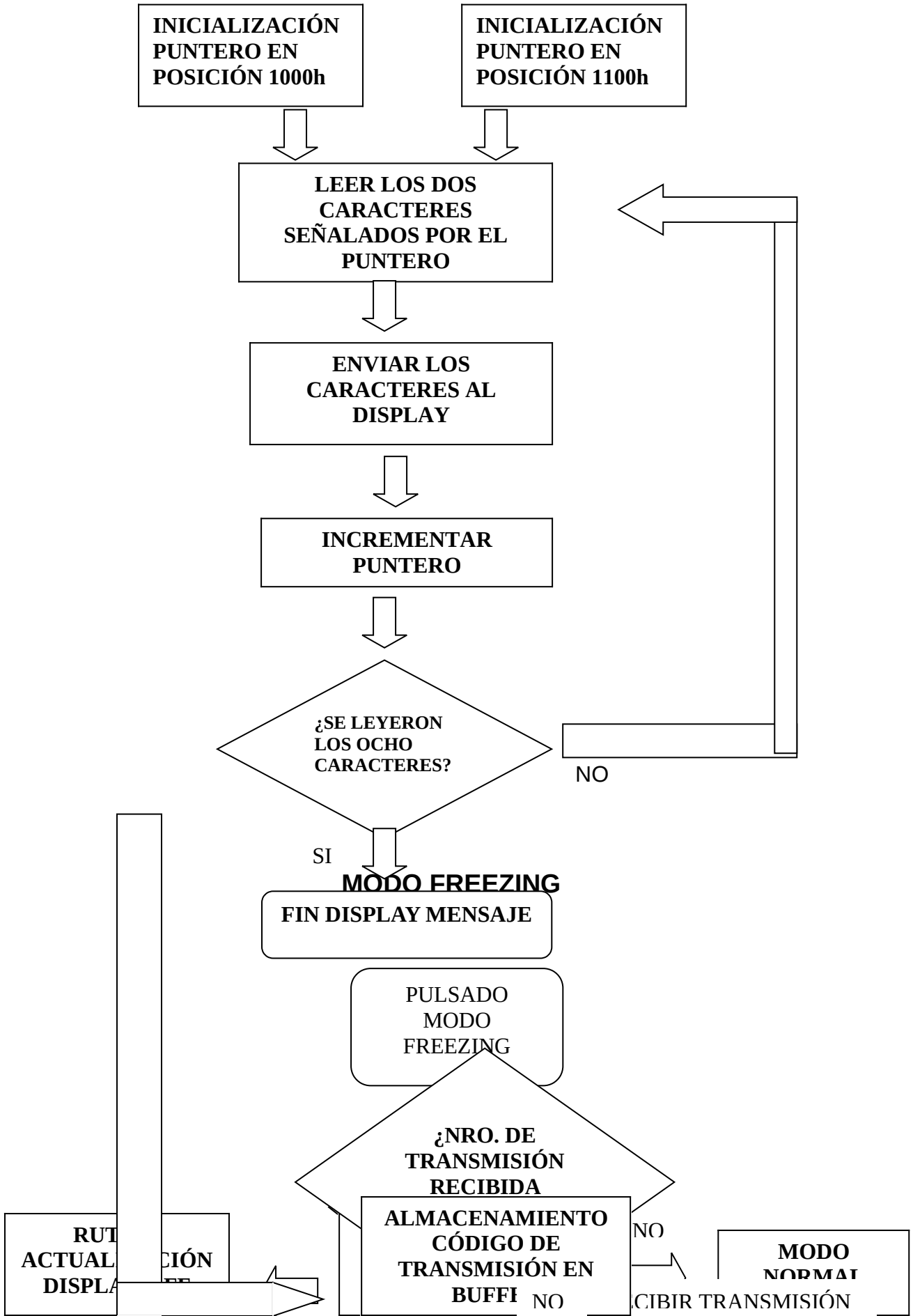


RUTINA DE DISPLAY



RUTINA CONFIGURACIÓN DE CARACTERES




INICIALIZACIÓN PUNTERO EN POSICIÓN 1000h
INICIALIZACIÓN PUNTERO EN POSICIÓN 1100h
LEER LOS DOS CARACTERES SEÑALADOS POR EL PUNTERO
ENVIAR LOS CARACTERES AL DISPLAY
INCREMENTAR PUNTERO
¿SE LEYERON LOS OCHO CARACTERES?
NO
SI
MODO FREEZING
FIN DISPLAY MENSAJE
PULSADO MODO FREEZING
¿NRO. DE TRANSMISIÓN RECIBIDA
ALMACENAMIENTO CÓDIGO DE TRANSMISIÓN EN BUFFI
NO
MODO NORMAL
NO

3.2.2.1 ULTIMAS MODIFICACIONES DEL PROGRAMA RECEPTOR

La implementación práctica sobre los sistemas de comunicaciones en VHF/UHF preexistentes, obligaron a sucesivos ensayos y modificaciones del software del sistema receptor.

1. Para lograr una baja sensibilidad al ruido recibido por el receptor VHF/UHF con el squelch abierto, fue necesario generar mediante programa un detector de correlación de frecuencia para cada dígito / modo de transmisión detectado. Con esto, la falta de correlación entre detecciones sucesivas provoca el fin de la recepción, volviendo a la configuración previa. Esta subrutina fue incluída en la rutina de interrupción de input capture.

Del mismo modo se diseñó un filtro por software para rechazar posibles señales captadas fuera de rango interferentes en el sistema. La rutina de decisión se encarga de rechazar estas señales finalizando la recepción.

2. Se incluyó además una rutina de presentación en el display, para señalización del correcto funcionamiento del receptor.

3. Se efectuó el agregado de un pulsador de borrado del display, como opción disponible para el usuario.

Todas estas modificaciones pueden observarse en el programa del código fuente del receptor (Apéndice C – Sección C.2.2).

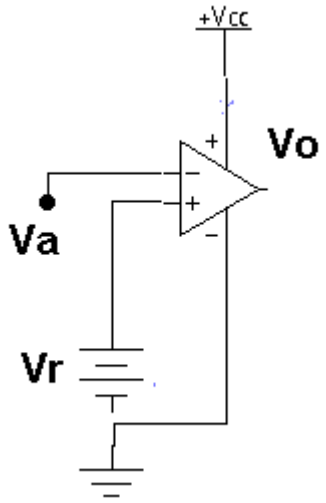
3.2.3 CIRCUITO DE ENTRADA DEL RECEPTOR

Como se vio anteriormente, el canal de transmisión introduce distorsión en la señal transmitida. Es necesario entonces reconstruir la señal original de salida del transmisor. Para ello, se emplea un circuito con filtro, amplificador, rectificador y comparador que a partir de la señal distorsionada obtiene una cuadrada de frecuencia idéntica a la original. Esto es posible gracias a que las frecuencias fundamentales se mantienen por el ancho de banda de transmisión utilizado.

La función del filtro de entrada es atenuar el ruido, (en tanto que el amplificador lleva el nivel de señal a uno mayor al de sensibilidad del comparador). El rectificador es el encargado de lograr el formato unipolar de la señal, dentro del rango de tensiones usados en el comparador.

SISTEMA COMPARADOR

El sistema comparador diseñado usa un LM339N. Fue diseñado con histéresis para ser más inmune al ruido. A continuación se realiza un análisis teórico para su mejor comprensión.



El comparador más simple es el implementado a través de un amplificador operacional y su funcionamiento es muy sencillo. Dada la figura 1, si Va> Vr, la salida Vo será Vcc, mientras que para Va<Vr, Vo =0V.

Figura 1

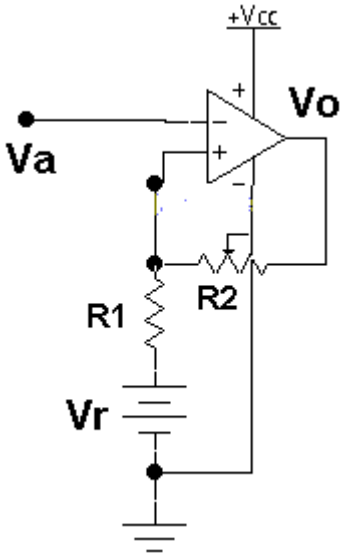


Figura 2

La idea de un comparador con histéresis es agregarle al comparador anterior, una realimentación positiva. De esta manera, la ganancia a lazo abierto se modifica, logrando una pendiente más abrupta (ver fig.3). Dada la figura 2 , a medida que se va disminuyendo el valor de R2, se tiene cada vez más realimentación positiva, hasta llegar al límite en que la pendiente del comparador se hace prácticamente vertical. En ese caso, se está en presencia de Ao.β=1 (fig. 4). Si se disminuye aún más R2, Ao.β>1 y se produce el fenómeno de histéresis, con un gráfico de ganancia como el que se ve en la fig.5

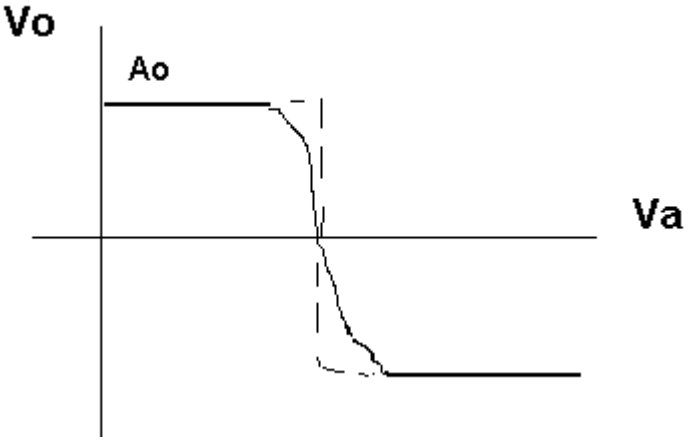
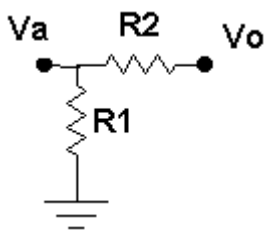


Figura 3



$$B = \frac{R1}{R1 + R2}$$

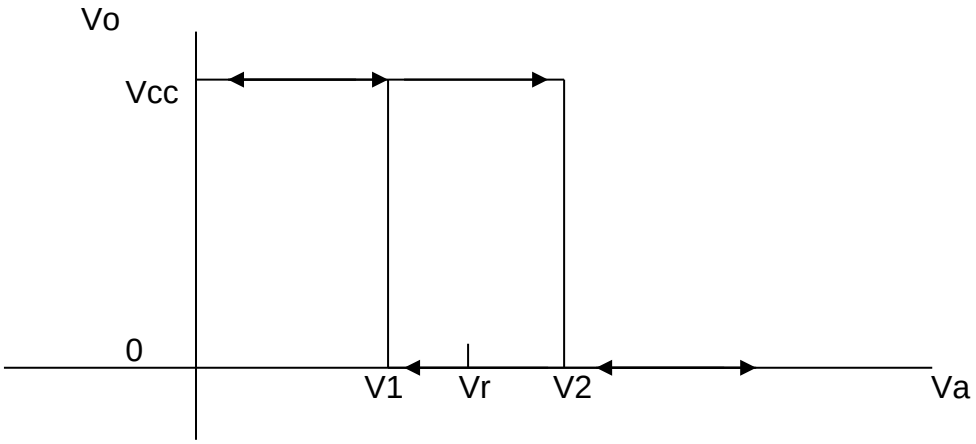
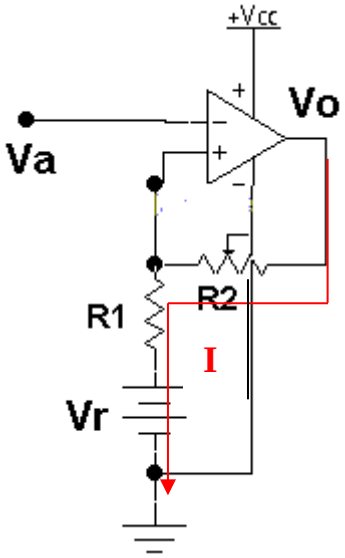


Figura 5

La aplicación del comparador con histéresis es la siguiente:

Dada una señal de entrada con ruido inicialmente con un nivel inferior a V1 , hasta que la señal no llegue al nivel superior de disparo V2, la salida no va a cambiar. Suponiendo que en un momento dado alcance este nivel V2, el ruido no lo podrá regresar a la condición inicial porque para que se produzca un nuevo disparo, el nivel debería bajar recién hasta V1.

Se verá ahora el cálculo teórico de dichos niveles de disparo.



Supóngase la entrada Va en estado bajo, con lo que la salida Vo está en estado alto. Si la salida se encuentra en alto, habrá entonces una corriente hacia Vr (fig. 6)

La tensión en la entrada positiva será:

$$V+ = Vr + I.\,R1 = Vr + \frac{Vcc-Vr\,.\,R1}{R1+R2}$$

Si se va aumentando el nivel de la entrada Va, recién cuando la entrada supere Vr más la caída en R1 se va a producir el cambio de nivel en la salida. Ese será entonces, el nivel de disparo superior:

$$V2 = Vr + I.R1 = Vr + \frac{Vcc - Vr}{R1+R2} . R1$$

Si ahora la salida es 0V, la corriente I será ahora en sentido contrario y para que se produzca un nuevo vuelco en la salida, Va tendría que bajar por debajo de:

$$V1 = Vr - I.R1 = Vr - \frac{Vr . R1}{R1+R2}$$

Como se ve en la fig. 7

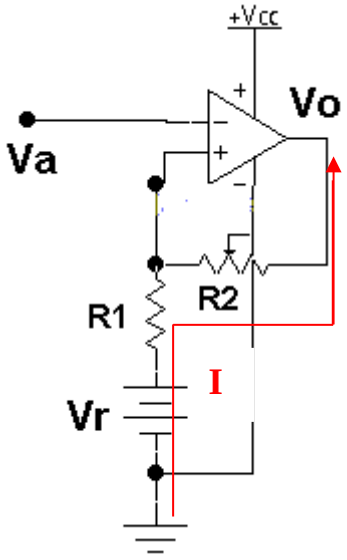
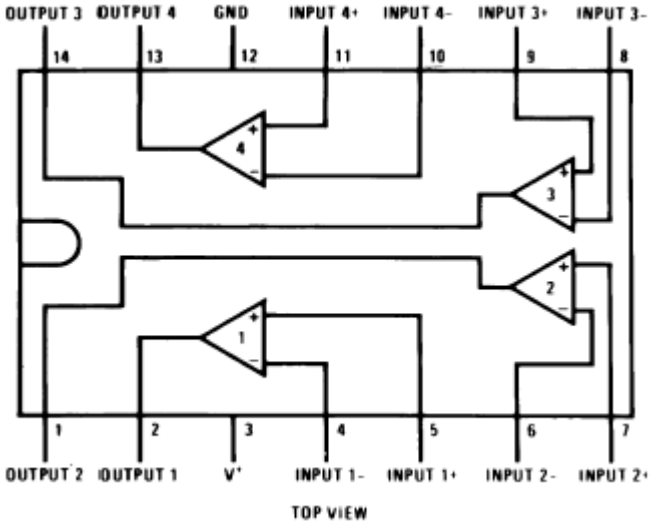


Figura 7

3.2.3.1 IMPLEMENTACIÓN DEL COMPARADOR EN EL LM339

El LM339N consiste en cuatro comparadores de tensión independientes y de precisión El correspondiente diagrama de conexión es:



El circuito equivalente del comparador LM339 para fuente simple puede verse en la siguiente figura:

El circuito desarrollado con histéresis es de la forma:

Donde : R1 = 10KΩ RL = 1KΩ

R2 = 10KΩ RH =100KΩ

3.2.4 DISPLAY LCD

Este capítulo describe las características del Display de Cristal Líquido (LCD) empleado en nuestro proyecto final. Primero se mencionan algunos aspectos generales para luego estudiar cómo se controla el mismo. Finalmente mostraremos el diagrama en bloques de inicialización del display.

Características Generales

El display de carácter empleado es el WM-C0802M-1YLYb de Wintek, de dos líneas de ocho caracteres cada una. Cada carácter dispone de una grilla de 5x7 pixels con un cursor por debajo. Los distintos modelos de la serie pueden identificarse como se ve en la siguiente figura

Pág 4 del manual

El WM-C0802M de Wintek funciona con +5V y consume una corriente típica de 2,5 mA. Tiene un controlador incorporado que lo convierte en un display llamado inteligente (SMART LCD MODULE). El controlador es un ST7066 de Sitronix o un equivalente de otra empresa.

Una de las razones principales de la elección de este tipo de display LCD (Smart LCD) es que presenta grandes ventajas respecto al resto:

► Tiene una interfase sencilla de implementar con prácticamente cualquier MCU

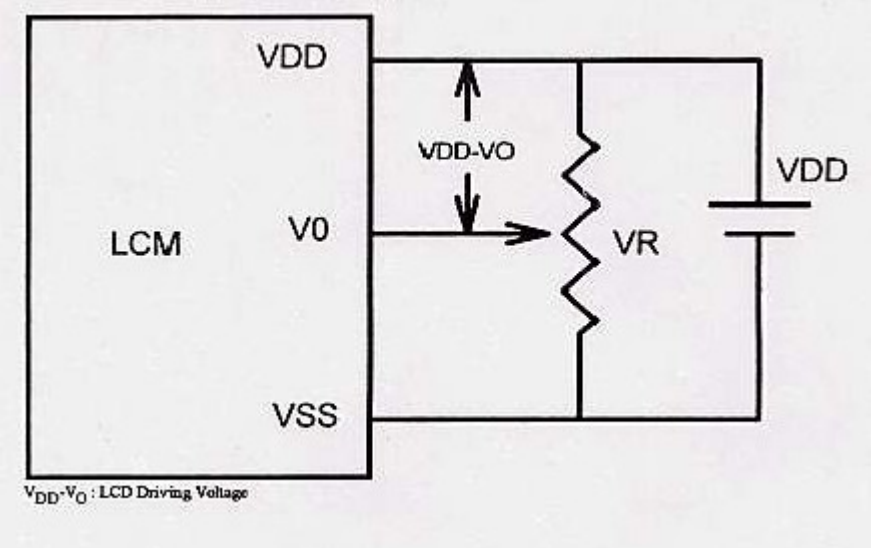
► Requiere menos software y Hardware para su control.

Si bien no todas son ventajas, ya que este método es un poco más costoso y está limitado por las capacidades del módulo LCD, resulta más cómodo en su aplicación

Interfase

La interfase con el exterior se realiza a través de un bus de datos de ocho bits (DB0 / DB7) y tres líneas de control (Entradas). Una sirve para habilitar el dispositivo (E), otra para la selección de registro (Instrucciones ó datos) denominada RS, y la tercera para indicar sinla operación a realizar será de escritura o lectura (R/-W)

Además de las líneas de alimentación VDD y Vss, el display requiere una línea adicional Vd para manejar el ajuste de contraste. Para una fuente de alimentación simple se lo conecta de la siguiente forma



A continuación se muestra un cuadro con las funciones de cada pin de interfase:

NO	SYMBOL	LEVEL	FUNCTION
1	V_{SS}	-	GND (0 V)
2	V_{DD}	-	VCC (+5 V ± 5%)
3	V_0	-	CONTRAST ADJUSTMENT
4	RS	H/L	REGISTER SELECT SIGNAL
5	R/W	H/L	READ/WRITE SELECTION
6	E	H,H→L	ENABLE SIGNAL
7	DB 0	H/L	DATA BIT 0
8	DB 1	H/L	DATA BIT 1
9	DB 2	H/L	DATA BIT 2
10	DB 3	H/L	DATA BIT 3
11	DB 4	H/L	DATA BIT 4
12	DB 5	H/L	DATA BIT 5
13	DB 6	H/L	DATA BIT 6
14	DB 7	H/L	DATA BIT 7

DIAGRAMA EN BLOQUES

Para entender el funcionamiento del Display LCD resulta útil analizar rápidamente el diagrama en bloques de las distintas partes del módulo:

Registro de Datos (DR): es usado para el almacenamiento temporal de los datos de lectura/escritura desde o hacia las memorias DDRAM y CGRAM.

Registro de Instrucciones (IR) se encuentra disponible para almacenar los códigos de instrucción e información de direcciones de las memorias DDRAM y CGRAM.

El Busy Flag (BF) indica si el módulo puede o no aceptar la próxima instrucción. En este sentido, si se encuentra en “uno” lógico, significa que el módulo está realizando una operación interna y no puede aceptar nuevas instrucciones. Existen consideraciones de tiempo para este flag que se desarrollarán en la sección siguiente.

CGROM (Carácter Generator ROM) Esta memoria ROM genera 192 patrones de caracteres de 8 bits.

CGRAM (Carácter Generator RAM) permite al usuario re-escribir patrones de caracteres libremente de acuerdo con el programa. Se desarrollará un poco más adelante junto con la memoria CGROM

Contador de Direcciones (AC) es usado para dar la información de las direcciones DDRAM y CGRAM.

DDRAM (Display Data RAM) es usada para guardar los datos del display expresados en códigos de carácter de 8 bits. En ella pueden ser almacenados, como se verá luego, hasta 80 caracteres.

Y finalmente un Circuito de Control de Cursor y Parpadeo

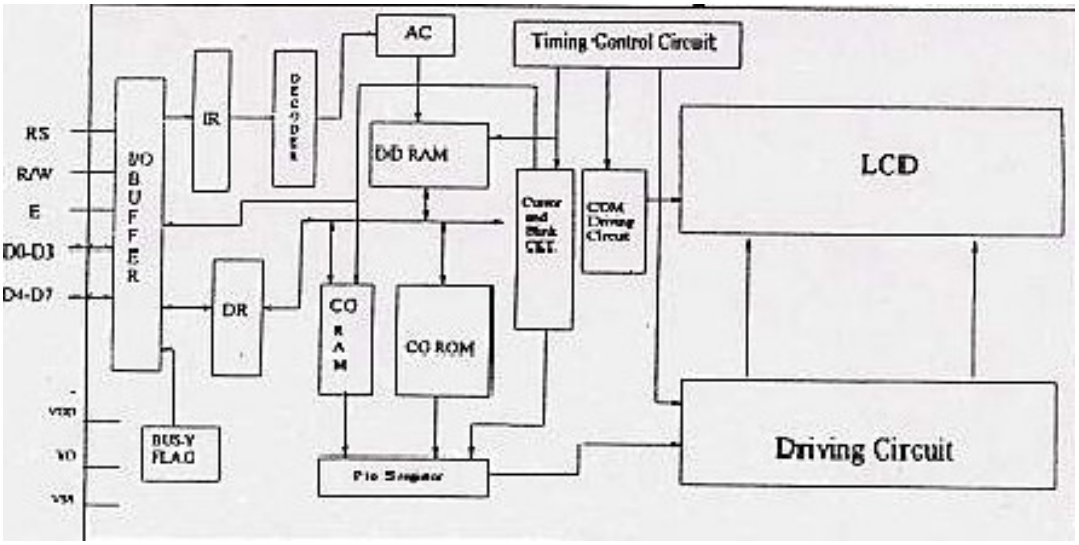


Diagrama en Bloques

Diagramas de Tiempo

Con el flanco ascendente de entrada E, el display registra las líneas de entrada RS y R/-W en un latch. En el caso de tratarse de una escritura registra los datos con el flanco descendente de la entrada E tal como se ve en la siguiente figura.

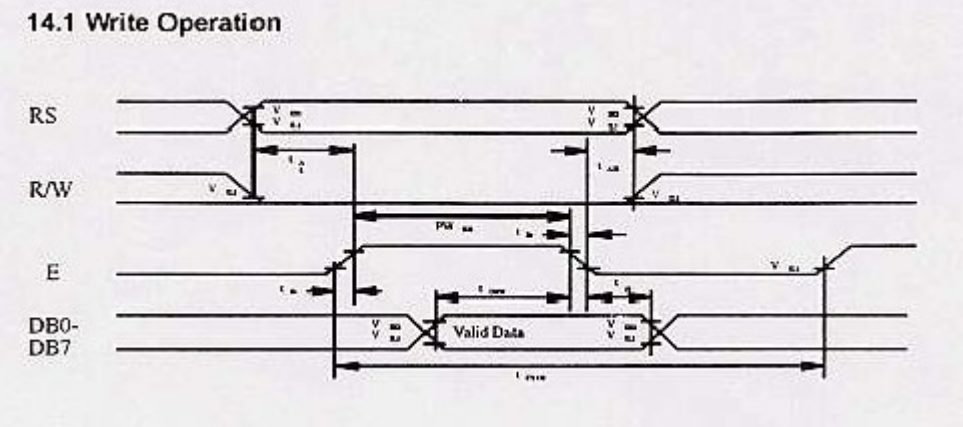


Diagrama de Tiempo para la escritura

Deben verificarse además los tiempos de escritura del siguiente cuadro:

(Writing data from MPU to LCM)

Item	Symbol	Limit (Min.)	Limit (Max.)	Unit
Enable Cycle Time	t_{CYCE}	666	-	nS
Enable Pulse Width (High level)	PW_{EH}	300	-	nS
Enable Risc/Fall Time	t_{Er}, t_{Ef}	-	25	nS
Address Set-Up Time (RS,R/W, E)	t_{AS}	100	-	nS
Address Hole Time	t_{AH}	10	-	nS
Data Set-Up Time	t_{DSW}	100	-	nS
Data Hold Time	t_H	10	-	nS

Cuadro de Tiempos de Escritura en el LCD

Cuando se trata de una operación de lectura, los datos serán válidos desde un pequeño tiempo después del flanco ascendente hasta un pequeño tiempo después del descendente.

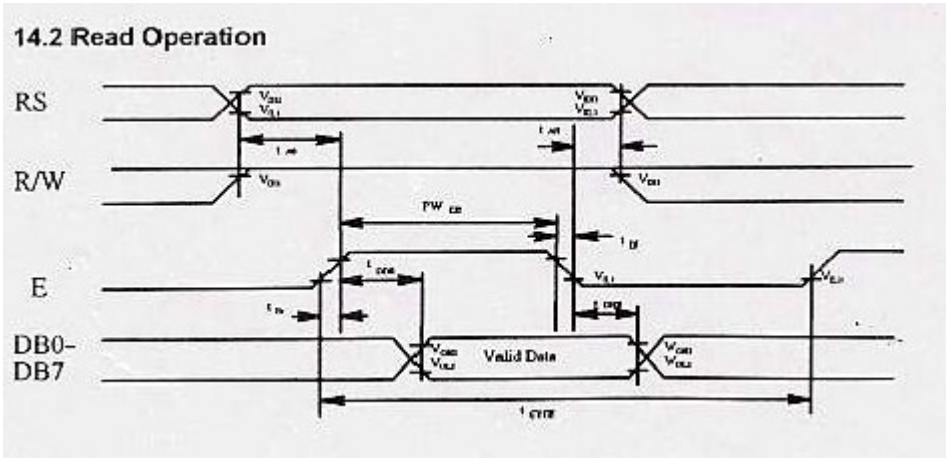


Diagrama de Tiempos para la Lectura

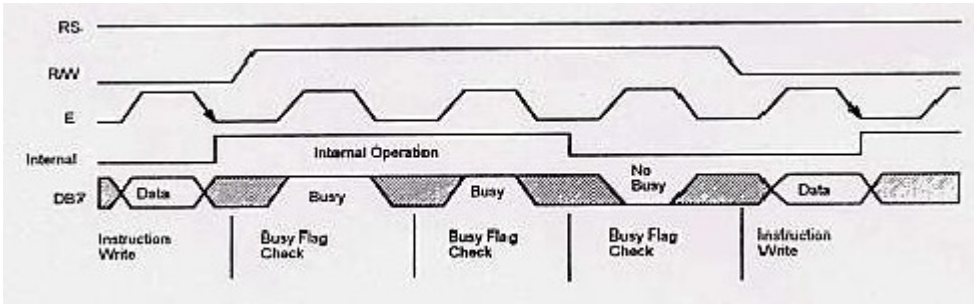
Deben verificarse además los tiempos de lectura del siguiente cuadro:

(Reading data from LCM to MPU)

Item	Symbol	Limit (Min.)	Limit (Max.)	Unit
Enable Cycle Time	t_{CYCE}	666	-	nS
Enable Pulse Width (High level)	PW_{EH}	300	-	nS
Enable Rise/Fall Time	t_{Er}, t_{Ef}	-	25	nS
Address Set-Up Time (RS,R/W, E)	t_{AS}	100	-	nS
Address Hole Time	t_{AH}	10	-	nS
Data Delay Time	t_{DDR}	-	190	nS
Data Hold Time	t_{DHR}	20	-	nS

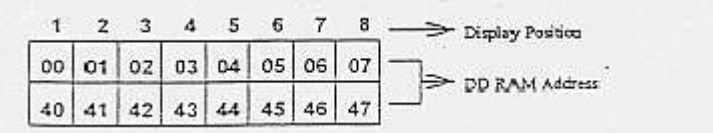
Lectura del Busy Flag

Es necesario antes de enviar la instrucción siguiente, realizar el chequeo del BF, para lo cual se deben esperar dos flancos negativos del DB7 tras los cuales se podrá efectuar una nueva operación. El diagrama de tiempos es el que sigue:



Memoria DDRAM

Cada posición de un carácter tiene asignado una posición de una memoria llamada DDRAM. En la siguiente figura está graficado el mapa de esta memoria:



Mapa de Memoria DDRAM

Las posiciones de los caracteres son 00h a 07h para la primer línea y 40h a 47h para la segunda línea. Sin embargo, es posible desplazar la información presentada hacia la derecha o la izquierda pudiendo visualizar en la primer línea ocho caracteres desde 00h a 27h y la segunda desde 40h a 67h.

MEMORIA CGRAM / CGROM

Las posiciones de memoria DDRAM definen los lugares en los que van a mostrarse los sucesivos caracteres. Dichos caracteres están determinados por una dirección de la llamada memoria CGRAM / CGROM. La memoria CGROM contiene la configuración de pixels de ciertos caracteres predefinidos en código ASCII. Por otro lado, la memoria CGRAM da la posibilidad de incluir caracteres definidos por el usuario, para la cual existen ocho posibles direcciones. A continuación se incluye el mapa de la memoria CGRAM / CGROM

Mapa de Memoria CGRAM / CGROM

COMANDOS

Ahora se verán todas las posibles operaciones sobre el display , resumidas en el siguiente cuadro

18. Instruction Set

Instruction operation

Function	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execu. Time* (Max.)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and return the cursor to home position (address 0).	1.64mS
Return Home	0	0	0	0	0	0	0	0	1	×	Return the cursor to the home position. Also returns the display being shifts to the original position. DD RAM contents remain unchanged. Set DD RAM address to zero.	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Set cursor move direct and specifies or not to shift the display. These operations are performed during data write/read of DD RAM/CG RAM. For normal operation, sets S to zero. I/D=1; increment; I/D=0; decrement S=1; accompanies display shift when data is written. for normal operation, set to zero.	40µS
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Set ON/OFF all display (D), cursor ON/OFF (C), and blink of cursor position character (B). D=1: ON display, D=0: OFF display. C=1: ON Cursor, C=0: OFF cursor, B=1: ON Blink Cursor, B=0, OFF Blink Cursor	40µS
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	×	×	Move the cursor and shift the display without changing DD RAM contents. S/C=1: Display Shift, S/C=0: Cursor move, R/L=1: shift to right, R/L=0: shift to left.	40µS
Function Set	0	0	0	0	1	DL	N	F	×	×	Set the interface date length (DL), number of display lines (N) and character font (F). DL=1: 8 bits, DL=0: 4 bits N=1: 2 lines, N=0:1 lines, F=1: 5×10 dots, F=0: 5 ×7 dots	40µS
Set CG RAM Addr.	0	0	0	1	ACG						Set CG RAM address. CG RAM data is sent and received after this setting.	40µS
Set DD RAM Addr.	0	0	1	ADD							Set DD RAM address. DD RAM data is sent and received after this setting.	40µS
Read busy flag & Addr	0	1	BF	AC							Reads BUSY FLAG (BF) indicating internal operation is beginning performed and reads address counter contents. BF=1: internally operating. BF=0: can accept instruction.	0µS
Write Data to CG RAM	1	0	WRITE DATA								Write data into DD RAM or CG RAM.	40µS**
Read Data from CG/DD RAM	1	1	READ DATA								Read data from DD RAM or CG RAM	40µS**

* 1. When f_{CP} or f_{OSC} is 250KHz.

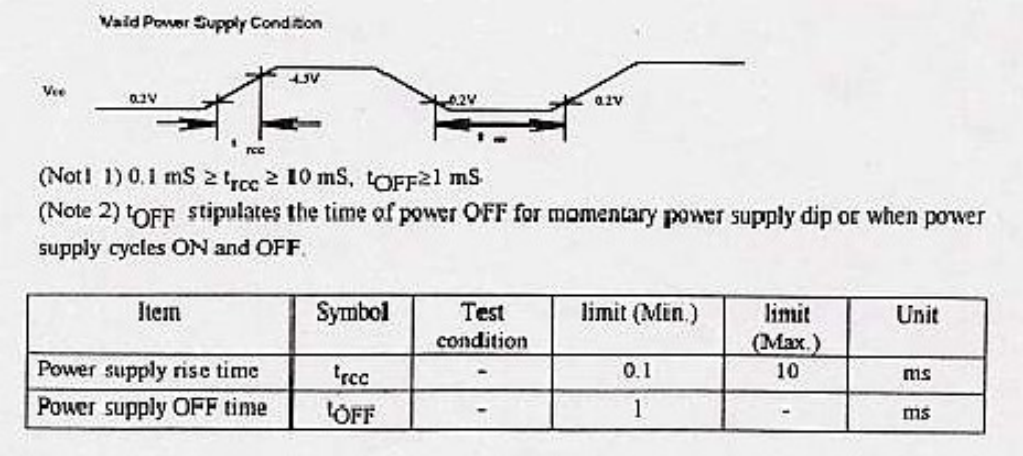
2. Execution time changes when frequency changes. When f_{CP} or f_{OSC} is 270KHz: $40\mu S \times \frac{250}{270} = 37\mu S$

** t_{ADD}=6µS

INICIALIZACION DEL DISPLAY

La inicialización del display puede realizarse por dos caminos:

Puede ser automática si se cumplen las condiciones de alimentación para el correcto funcionamiento del circuito interno de reset;, que se detallan a continuación:



(Not1 1) 0.1 mS ≥ t_{rcc} ≥ 10 mS, t_{OFF}≥1 mS

(Note 2) t_{OFF} stipulates the time of power OFF for momentary power supply dip or when power supply cycles ON and OFF.

Item	Symbol	Test condition	limit (Min.)	limit (Max.)	Unit
Power supply rise time	t_{rcc}	-	0.1	10	ms
Power supply OFF time	t_{OFF}	-	1	-	ms

De otro modo, es necesaria una inicialización por instrucciones siguiendo el procedimiento del diagrama de flujo de la página siguiente:

Especificaciones Importantes

Es importante mencionar que el consumo de corriente resulta del orden de los 1,6 mA (típico) hasta unos 2,5 mA máximos (para el caso en que todos los leds se encuentren prendidos).

En el Apéndice se detallan el resto de las hojas de datos del modelo usado , así como un programa de prueba.

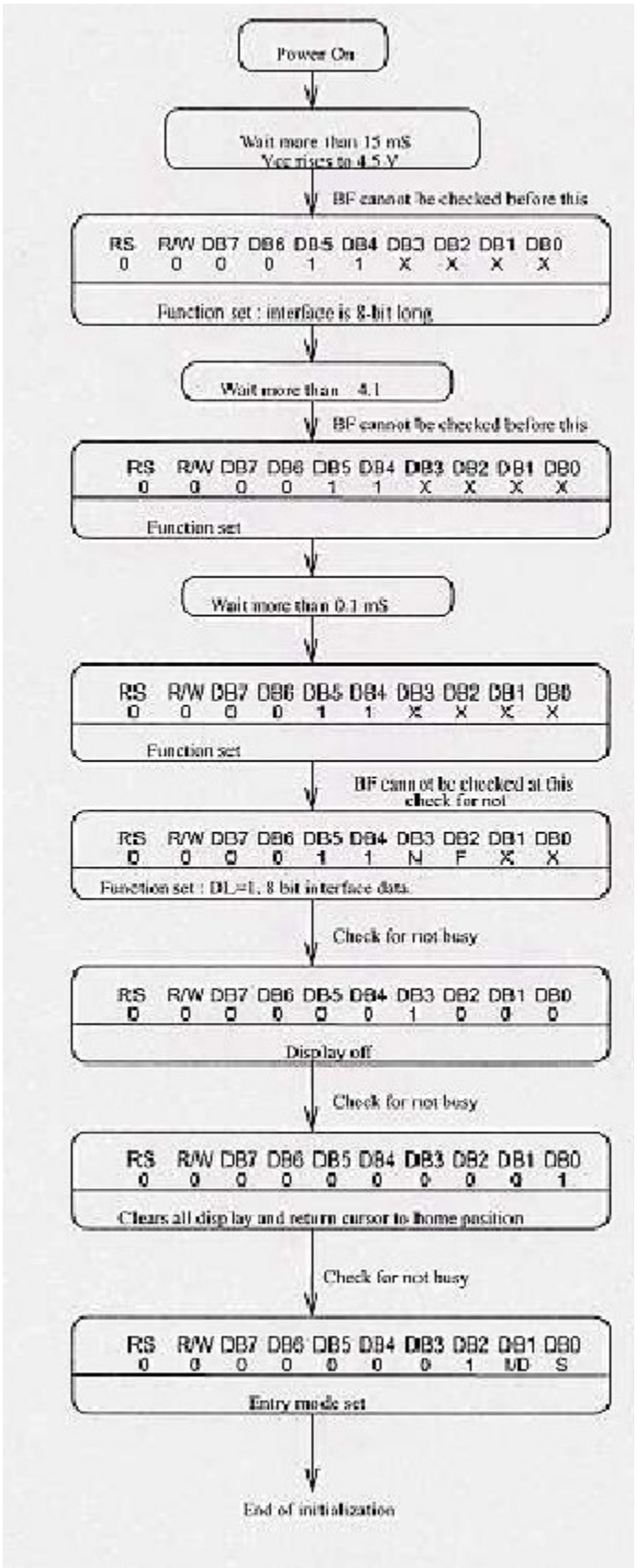


Diagrama de flujo de Inicialización

3.3 PROTOCOLO EMPLEADO

INTRODUCCIÓN

La transmisión de los códigos debe cumplir con una normativa de telecomunicaciones. Dichas normas o standards cumplen una triple misión fundamental: obtener una interoperabilidad entre los distintos sistemas de comunicaciones existentes, asegurar un alto grado de calidad en su funcionamiento y lograr consistencia en la evolución de estos sistemas.

Para la regulación de estas normas surgió el ITU (International Telecommunication Union), organismo dentro del cual se encuentra el ITU-R, llamado anteriormente CCIR (Comité Consultatif Internationale de Radio), que se ocupa del sector de radiocomunicaciones. El rol principal de este subcomité es regular la asignación de radiofrecuencias y así reducir la interferencia entre las distintas estaciones de radio.

En este sentido, el protocolo utilizado -uno de los recomendados por el ITU-R- se encarga de definir el formato de los tonos de transmisión (Duración y Período del Tono) y las frecuencias y el rango de detección para los distintos dígitos usados. A continuación se muestran dos cuadros con las características de este protocolo.

Formato	Período del Tono (seg.)	Tiempo sin Tono (seg.)
EIA	0,033	0,060

Cuadro 1. Características del Formato de los Tonos

Tono No.	Frecuencia del Tono		
	Codificación	Decodificación	
		Mínima	Máxima
0	1981	1941	2024
1	1124	1100	1148
2	1197	1171	1222
3	1275	1249	1301
4	1358	1328	1386
5	1446	1417	1477
6	1540	1509	1571
7	1639	1605	1673
8	1747	1712	1785
9	1860	1822	1900
A	2401	2352	2454
B	930	911	950
C	2246	2200	2291
D	991	971	1012
E	2109	2064	2154

Cuadro 2. Características de Frecuencia para los códigos

CÁLCULO DE LOS VALORES PARA EL REGISTRO TCH0 (TIMER CHANNEL 0)

A partir de las frecuencias estipuladas en el protocolo, es necesario calcular los valores con los que se debe cargar el registro contador de la función 'Output Compare'. Como ya se ha mencionado, todos los registros del timer operan a la cuarta parte de la frecuencia del cristal dividido por la opción de prescaler configurada. Se debe considerar asimismo, el ciclo de trabajo utilizado. En términos generales, esto se puede representar bajo la ecuación:

$$N = f_{oper} * T_{tono} * \text{Ciclo Trabajo} = \frac{f_{XTAL} * T_{tono} * \text{Ciclo Trabajo}}{4 * PS}$$

donde:

N es el número que se debe guardar en el registro TCH0

f_{oper} es la frecuencia de operación del microcontrolador

T_{tono} es el período del tono a transmitir

PS es la opción de prescaler establecida

Dada la configuración usada en el programa (ver Cap.), en este caso, el cristal tiene una frecuencia de oscilación de 10 MHz., el ciclo de trabajo es del 50% y no se hizo uso de la opción de prescaler, obteniendo la siguiente tabla:

Tono Nro.	Número
0	631
1	1112
2	1044
3	980
4	920
5	864
6	812
7	763
8	716
9	672
A	521
B	1344
C	557
D	1261
E	593

Finalmente, se puede hacer un cálculo análogo al anterior para calcular los valores umbrales para la detección según el protocolo utilizado. Estos valores se muestran en la Tabla 2.

Tono Nro.	Límite Superior	Límite Inferior
0	644	618
1	1136	1089
2	1067	1023
3	1001	961
4	941	902
5	882	846
6	828	796
7	779	747
8	730	700
9	686	658
A	531	509
B	1372	1316
C	568	546
D	1287	1235
E	606	580

Tabla 2. Umbrales para la detección de cada tono

3.4 MÉTODO DE SINCRONIZACIÓN

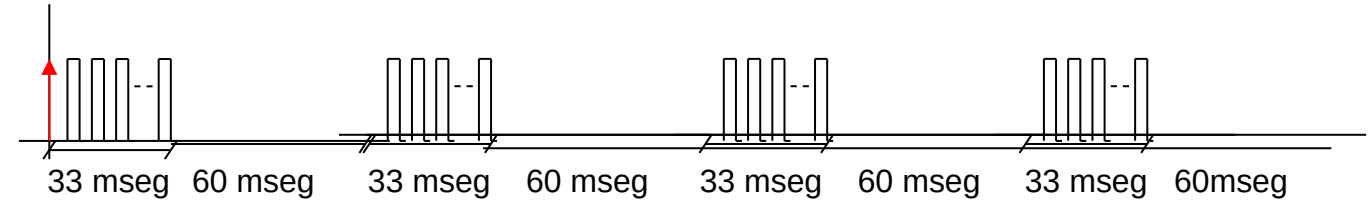
Una de las consideraciones más importantes para lograr una correcta detección es conseguir que el sistema receptor sense la entrada en los intervalos de tiempo en que debe hacerlo. De no ser así, podrían haber fallas por desplazamientos de tiempo en la lectura de los códigos.

Si bien, dado el formato de la señal dada por el protocolo -con cerca del doble de tiempo en estado fijo (arbitrariamente alto o bajo)- prácticamente se descarta la posibilidad de estar midiendo dos códigos sucesivos como si fueran el mismo; existen otros posibles inconvenientes tales como un posible transitorio no deseado que se produzca en el momento de la lectura de la entrada y que por un corrimiento pueda ocasionar una detección errónea. Es cierto que resulta una posibilidad casi nula debido principalmente al tiempo de trasmisión del código y a la cantidad de ciclos recibidos en este lapso (un mínimo de 30 ciclos).

Por lo mencionado anteriormente, si bien el formato del protocolo reduce notablemente los posibles errores, y el método de sincronización no es particularmente crítico, debe buscarse un método simple y que no afecte esencialmente los tiempos de comunicación. Las ideas al respecto son varias, pero la más sencilla y mayormente empleada es la sincronización por bits de start y stop. El bit de start no consiste más que en un flanco positivo de tensión, en tanto que el de stop es un flanco negativo

Otro posible mecanismo de sincronización es la transmisión de una frecuencia normalizada para definir el comienzo de la transmisión. Esta segunda posibilidad ofrecería más seguridad para la detección, dado que prácticamente anularía el efecto de cualquier señal interferente que podría llegar a la entrada del receptor. Sin embargo, la propia detección de esta frecuencia produciría un retardo en la trasmisión que, como se ve en las especificaciones resulta poco recomendable.

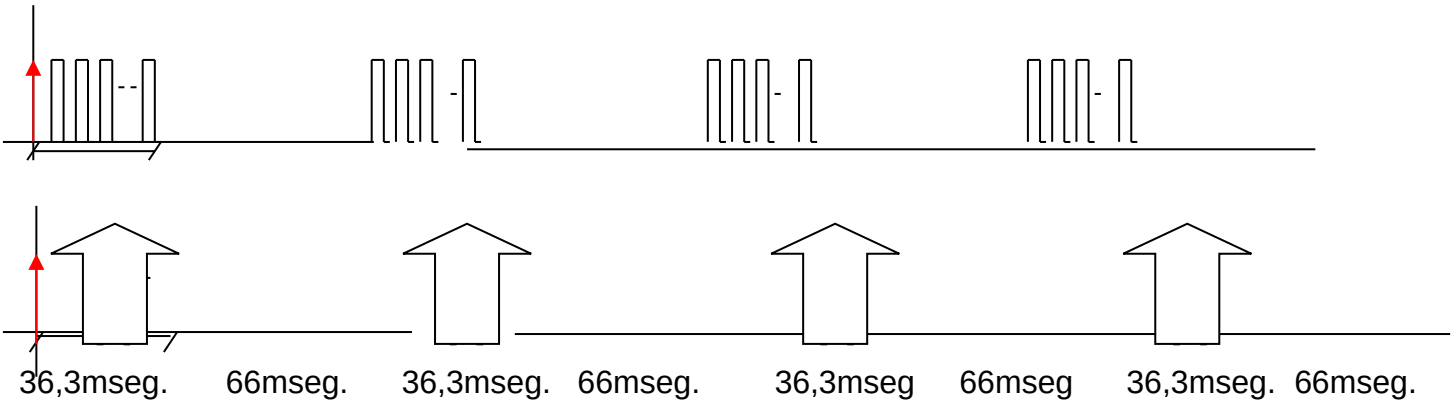
Se verán a continuación varios diagramas de tiempo a modo de visualizar mejor la cuestión y se analizará la problemática más detalladamente:



.

En principio, el hecho de usar una única sincronización para la comunicación de todos los códigos, disminuye el error de corrimiento únicamente al ocurrido en la detección del bit de start. Y el mayor error tolerable teórico es de alrededor de 31,8 mseg. (figura 1), con lo que es posible detectar un solo tono para el peor caso, o sea el tono de menor frecuencia (cerca de 900Hz.). Es cierto que en la práctica es necesario el sensado de varias frecuencias para un buen funcionamiento. Si consideramos unos quince tonos para lograr una detección correcta, el mayor error teórico tolerable pasaría a ser de unos 16 mseg.

Otro punto a analizar es que, si no utilizamos un mecanismo de sincronización para cada dígito transmitido, los errores son ahora debidos fundamentalmente a las diferencias entre las frecuencias de resonancia de los dos cristales empleados. En el mejor caso, el retardo en la lectura del bit de start se puede compensar en cierto modo con un cristal ínfimamente más rápido en el receptor que en el transmisor. En realidad, esto es relativo al grado de corrimiento tanto en las frecuencias de los cristales, como en la lectura del flanco inicial. Si es predominante el efecto de corrimiento en los cristales, El peor caso a analizar es que se sumen ambos errores, y al retardo en la lectura del bit de start se agregue el efecto de un cristal más lento en el receptor. En este caso, si sólo consideramos el efecto del corrimiento relativo del cristal receptor respecto del transmisor, tenemos que el máximo error tolerable puede verse en el gráfico de la figura 2:



Nota: las flechas indican los intervalos de sensado de los tonos

Y resulta aproximadamente en un error teórico máximo del 10% relativo al cristal transmisor. Nuevamente, esto es válido para la detección de un mínimo de un tono por dígito. En la práctica, con la misma consideración que se hizo anteriormente (15 tonos mínimos sensados por cada dígito), el error tolerable máximo es de alrededor de 0,75% relativo nuevamente al cristal transmisor.

ERRORES DE SINCRONIZACIÓN POR PROGRAMA

La idea de la programación del transmisor es enviar primero el flanco de comienzo de la transmisión y luego, tras la lectura del primer dígito y la configuración de registros, comenzar la comunicación propiamente dicha. Esto ocasiona una mínima pérdida de sincronización, puesto que el receptor comienza la comunicación apenas recibe el bit de start. Lo que ocurre es que al usar el Timer Overflow (Cap.) como instrumento de medición de los tiempos, la inicialización de los registros contadores del módulo del Timer de transmisor y receptor debe ser idealmente simultánea. Este error es en la práctica despreciable, pero puede ser disminuido, con el envio del flanco inicial justo antes de la inicialización (reset) del registro contador del Timer en el transmisor.

CAPÍTULO 4
MANUAL DE OPERACIÓN

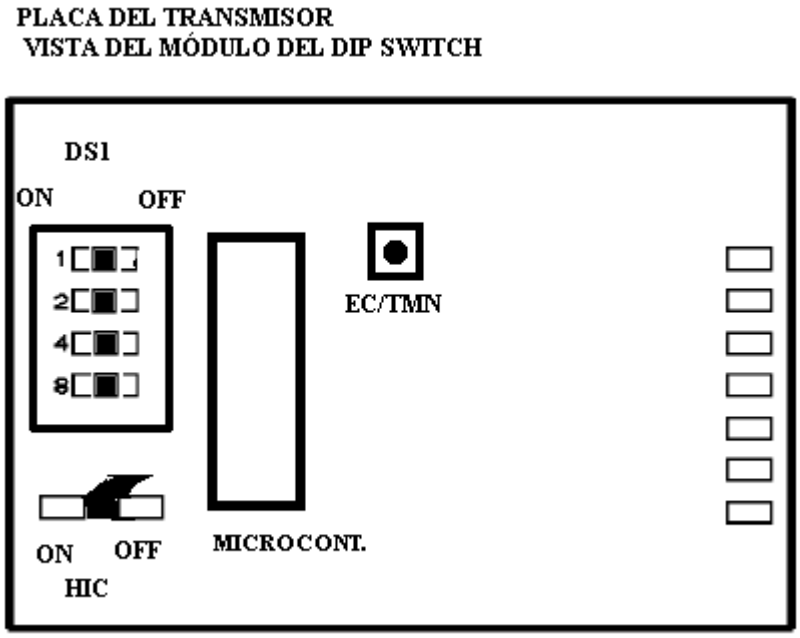
CAPÍTULO 4

MANUAL DE OPERACIÓN

El sistema se ha diseñado con un alto grado de flexibilidad para su manejo y relativa sencillez en su aplicación.

Se comienza por la configuración del código de trasmisión. Se plantean dos configuraciones distintas según los modelos posibles:

Configuración del modelo de Ingreso del Código por Dip-Switch



Para poder efectuar la programación del código debe, primero que nada, colocar la llave HIC (Habilitador Ingreso Código) en la posición ON, antes del encendido del equipo. De otra manera el código, que se trasmitirá será un código que se encuentra cargado en el microcontrolador.

Una vez realizado esto y encendido el equipo, se procede de la siguiente manera:

Dados los cuatro Dip-Switchs (DS1) en código binario en el esquema superior, se deben sumar cada uno de los valores asignados a los switchs (1, 2, 4 y 8) colocados en posición ON, para obtener el primer dígito del código deseado/se deben colocar en la posición ON aquellos switchs cuya suma de valores asignados (1, 2, 4 y 8) dé el primer dígito del código deseado.

. Luego, se debe pulsar EC/TMN (Entrada Código/ Transmisión Modo Normal) y procederse de manera idéntica con el segundo, tercer y último dígito respectivamente. A continuación se verá un ejemplo para facilitar la comprensión del procedimiento

Se debe ingresar el código 7643

Primero, se coloca la llave HIC en posición ON (figura 1)

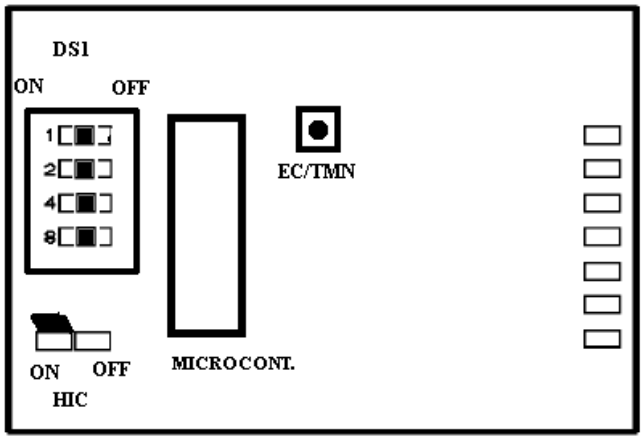
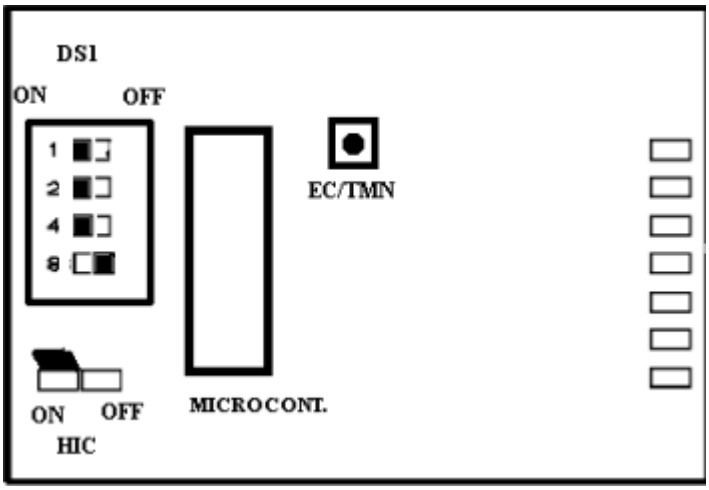


Figura 1

Segundo, se configuran los 4 switchs para obtener el primer código.En nuestro caso,

4+2+1=7 , por lo que los tres dip-switch superiores se colocan en ON y el cuarto en OFF, tal como se ve en la figura siguiente



Tercero, se entra el dígito seleccionado apretando el pulsador EC/TMN.

´Cuarto, se entra el segundo dígito del código en forma análoga al segundo paso. En este caso, 6=4+2, de modo que sólo estarán en posición ON estos dos switchs (figura 3)

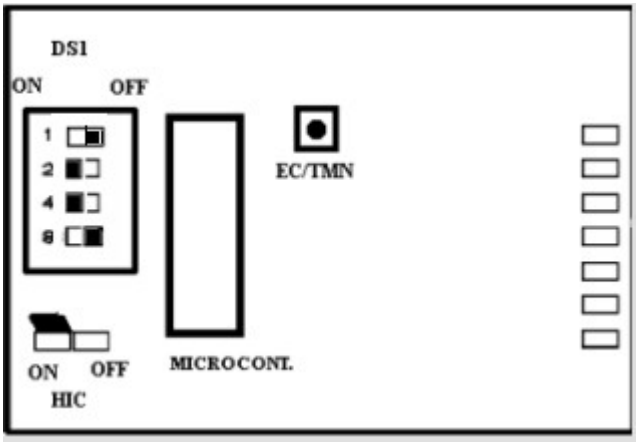


Figura 3

Quinto, se pulsa nuevamente EC/TMN

Sexto, se procede a configurar los dip-switchs para el tercer dígito del código. En este caso, sólo se coloca en ON el dip-switch con valor asignado 4 (figura 4) y se toca el pulsador para cargar el número.

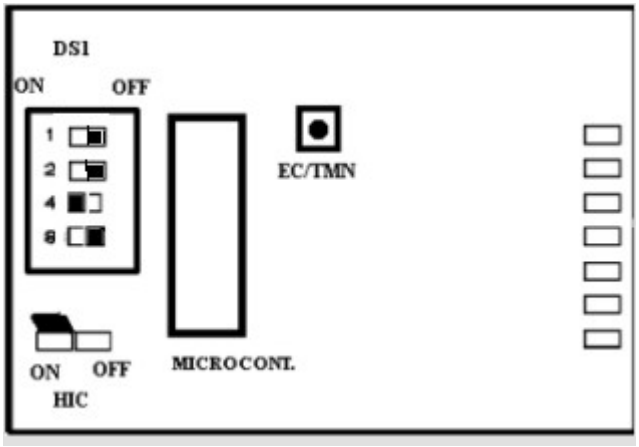
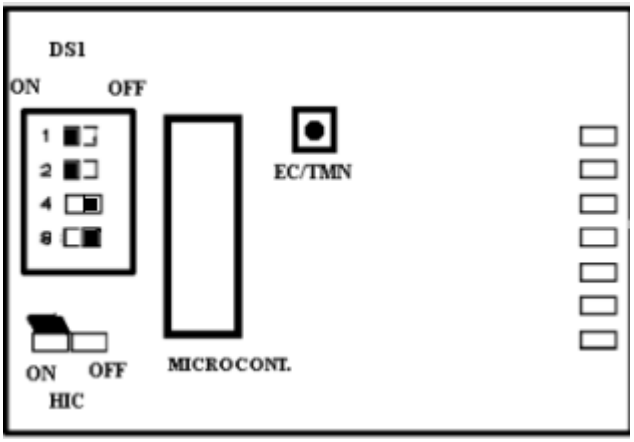


Figura 4

Séptimo, se programa el último dígito de igual manera. 3=1+2, por lo que deben colocarse en ON los dos switchs superiores (fig.5) y



Octavo y último, se pulsa EC/TMN para finalizar la carga del código.

Configuración del modelo de Ingreso del Código por Conversores A/D

Dado que este sistema trabaja en forma automática y simultánea para todos los dígitos del código, sólo son necesarios dos pasos para la configuración. Asimismo, es importante señalar que la configuración en este caso, se realiza antes de encender el equipo.

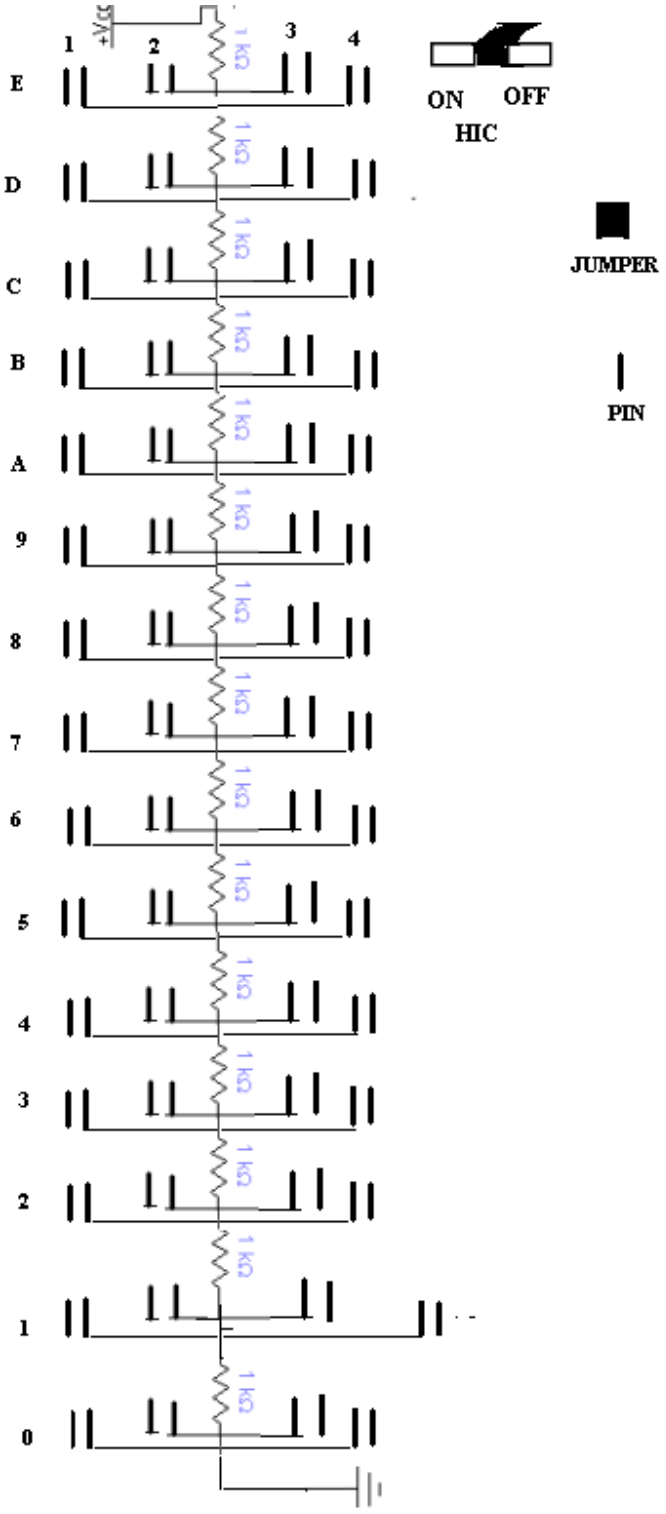
De igual forma que en el modelo anterior, la configuración comienza por la habilitación del ingreso del código, para lo cual debe colocarse el interruptor HIC en la posición ON.

El segundo paso es colocar los jumpers en los pines adecuados para obtener el código deseado, de acuerdo al criterio siguiente (ver figura 6 de la página siguiente): en el extremo superior del módulo se observan los nros. 1,2,3 y 4, que corresponden a cada dígito del código respectivamente. Dichos números indican las dobles columnas de pines que van entre los valores de 0 y E. Dichos valores, señalados en el extremo izquierdo del módulo son los valores posibles para cada dígito, de modo que para seleccionar un valor para cada dígito, basta con unir con jumpers los dos pines marcados en la intersección entre nro. dígito y valor de dígito

Veamos un ejemplo para clarificar el procedimiento

Supóngase se debe cargar el código 9628.

De acuerdo a lo visto, se comienza por colocar el interruptor HIC en la posición ON y configurar el módulo de acuerdo al criterio visto, obteniendo como resultado una configuración como la fig. 7



MODELO ESQUEMÁTICO DEL MÓDULO DE ENTRADA DE CÓDIGO POR CONVERSORES A/D

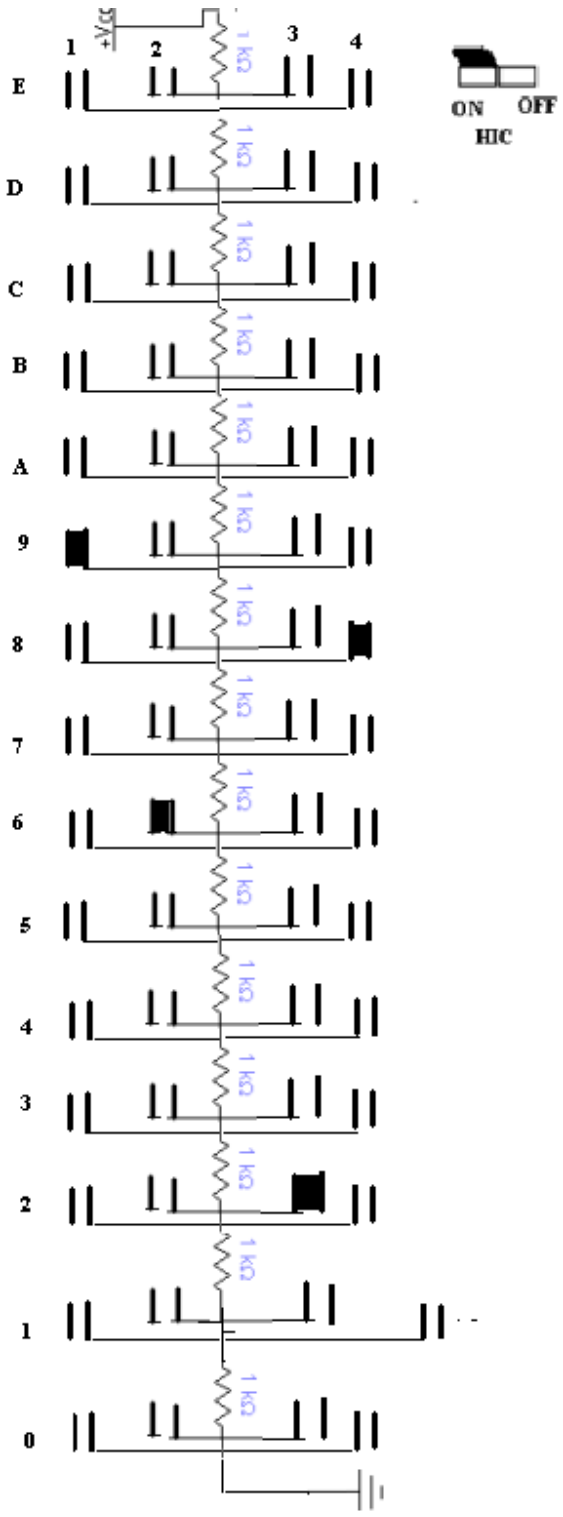


FIGURA 7. CONFIGURACIÓN RESULTANTE DE OBTENER EL CÓDIGO 9628

MODOS DE TRANSMISIÓN

Para la comunicación con la central, existen dos modos de transmisión principales:

Modo Normal

Modo de Emergencia

El Modo Normal es empleado para la identificación de la unidad móvil con la central y tiene una duración aproximada de 400 mseg. (Tiempo de envío de los cuatro dígitos del código de identificación del vehículo). Para dicha comunicación, se debe pulsar EC/TMN (Entrada Código / Transmisión Modo Normal). En el display de la Unidad Central aparecerá un mensaje como el siguiente:

#XXXX/SN

donde las X representan los dígitos del código del móvil que transmitió.
Y SN significa Status Normal

El Modo de Emergencia tiene una duración de unos 500 mseg. y se inicializa pulsando TME (Transmisión Modo Emergencia).Al recibir el Modo de Emergencia, en el receptor de la Unidad Central se mostrará en el Display :

XXXX/SE

donde las X representan los dígitos del código del móvil que transmitió.
Y SE significa Status Emergencia

En este caso, el display se congelará resaltando el código que se encuentra en este modo.

El resto de los módulos del equipo transmisor trabajan de manera automática y no requieren configuración alguna por parte del usuario.

CONFIGURACIÓN DEL RECEPTOR

El receptor actúa de modo automático una vez realizada la conexión entre sus módulos a saber:

Módulo Entrada

Módulo Receptor

Módulo Display

Finalmente, existe al acceso del usuario un pulsador F en el módulo Receptor, con el cual se podrá congelar el display instantáneamente, guardando las nuevas transmisiones en un bufer que permite almacenar hasta ocho códigos con sus modos de transmisión respectivos.

INTERFASE CON EL DISPLAY LCD

La interfase del microcontrolador receptor con el display LCD se realiza directamente mediante cable plano, siguiendo el interconexionado siguiente:

# PIN DISPLAY LCD	PIN MICROCONTROLADOR/ SALIDA
1	GND (MASA)
2	VCC (ALIMENTACIÓN 5V)
3	POTENCIÓMENTRO 10KΩ
4	PIN 3 PUERTO D
5	PIN 2 PUERTO D
6	PIN 5 PUERTO D
7	PIN 0 PUERTO B
8	PIN 1 PUERTO B
9	PIN 2 PUERTO B
10	PIN 3 PUERTO B
11	PIN 4 PUERTO B
12	PIN 5 PUERTO B
13	PIN 6 PUERTO B
14	PIN 7 PUERTO B

En tanto que también es posible realizar una conexión adicional para poder utilizar la opción de backlight que ofrece el display empleado.

CAPÍTULO 5
MANUAL DE MANTENIMIENTO

Capítulo 5

MANUAL DE MANTENIMIENTO

ESPECIFICACIONES

El equipo consta de un sistema transmisor y un receptor que cuenta con un display LCD a la salida.

El sistema transmisor está compuesto por los siguientes elementos:

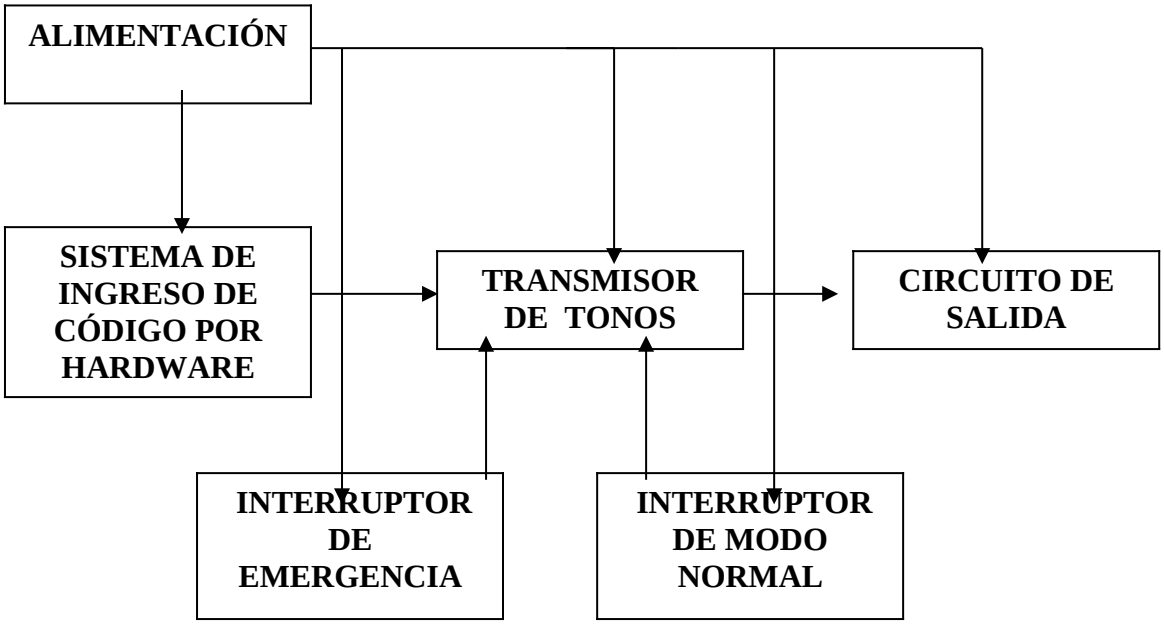
- Regulador de entrada 7805L
- Microcontrolador MC68HC908JK3
- Cuádruple Dip-Switch o Sistema de Entrada por Conversores ADC
- Oscilador Cristal 10MHz.
- Llave y Led de Encendido
- Dos Pulsadores
- Conexiones de Entrada
- Transistor C547B
- Potenciómetro 50KΩ
- Capacitores y Resistencias varias.

El sistema receptor está compuesto por los siguientes elementos:

- Regulador de entrada 7805L
- Filtro Pasabanda de Ancho de Banda de 3KHz.
- Circuito Comparador usando LM339N
- Microcontrolador MC68HC908JK3
- Oscilador Cristal 10MHz.
- Llave y Led de Encendido
- Pulsador
- Conexionado de Interfase con el Display LCD
- Display LCD Wintek WM-C0802M-1YLYb
- Dos Potenciómetros 10KΩ
- Capacitores y Resistencias Varias

DIAGRAMA EN BLOQUES

TRANSMISOR



Alimentación: la provee la batería del móvil

Sistema de Ingreso de Código por Hardware: está formado por un Dip-Switch Cuádruple o un sistema de entradas por los conversores A/D. Permite al usuario cargar con facilidad el código del móvil.

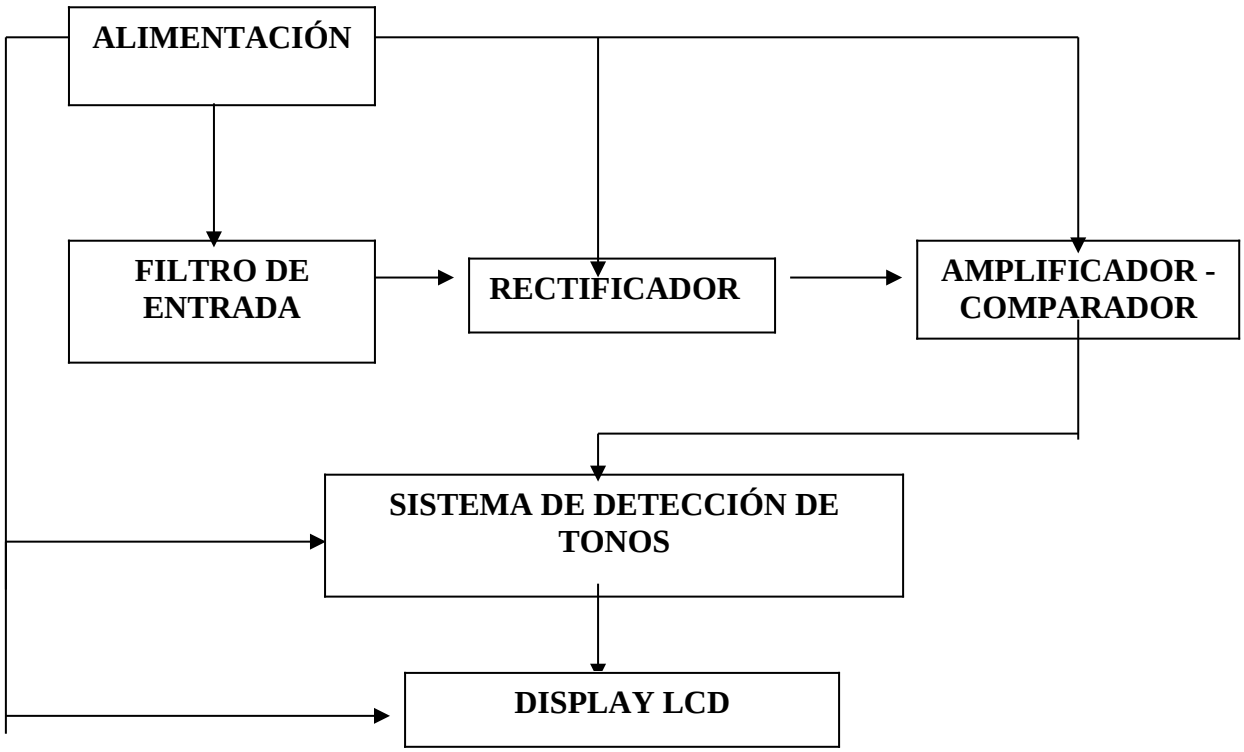
Transmisor de Tonos: está compuesto básicamente por el microcontrolador y es el encargado de generar todos los tonos con los que puede identificarse cada móvil.

Circuito de Salida: consta principalmente de un amplificador LM358N.y se encarga de adaptar la señal de salida a la entrada de modulación.

Interruptor de Emergencia: consiste en un pulsador que de ser activado, generará una transmisión en modo de Emergencia.

Interruptor de Modo Normal: consiste en un pulsador que de ser activado, generará una transmisión en modo Normal.

RECEPTOR



Alimentación: está provista por la red de 220 volts que llega a la central.

Filtro de entrada: Filtro Pasabajo de Tercer Orden Butterworth, cuyo fin es acotar el nivel de ruido y eliminar las señales fuera del rango deseado.

Rectificador: básicamente compuesto por un diodo, para llevar la señal dentro de los niveles de 0 a 5 volts.

Amplificador - Comparador: sistema constituido fundamentalmente por el integrado LM339N. Su misión es amplificar la señal de salida del rectificador y reconstruir la cuadrada original.

Sistema de Detección de Tonos: está constituido por el microcontrolador y se encarga de obtener el código del móvil.

Display LCD: sirve para visualización del código detectado y su correspondiente modo de transmisión.

PROCEDIMIENTO DE AJUSTE

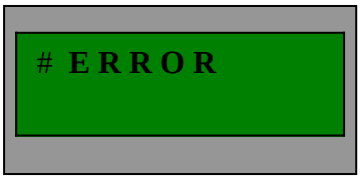
El sistema requiere pocos ajustes . Durante la instalación sólo hay que ingresar el código del móvil de modo correcto, tal como se detalla en el Manual de Operación. Los únicos ajustes necesarios en el receptor son el contraste del diaplay para lograr una visualización deseada y el l potenciómetro de offset de la señal de entrada para un correcto funcionamiento del comparador..

PROCEDIMIENTO DE LOCALIZACIÓN DE FALLAS

A. Encendido

Tanto el sistema transmisor como el receptor cuentan con un diodo led que indica el estado de encendido y apagado del equipo.

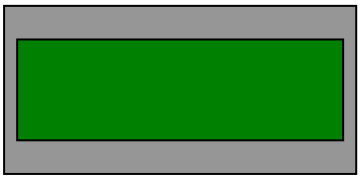
B. Mensaje de Error en el Display



De tener este mensaje en el display existen dos posibilidades: o bien, se efectuó un intento de comunicación fallido o bien, el receptor captó una señal interferente no deseada.

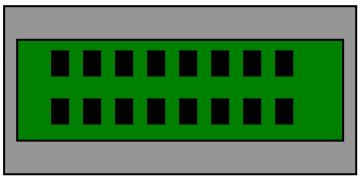
C. Visualización del Display

- Display en blanco



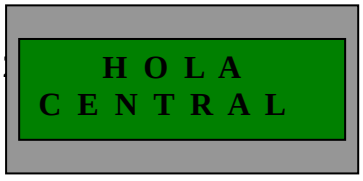
Este estado surge tras oprimir el botón de RESET DISPLAY. Si este no es el caso es necesario verificar la alimentación y ajustar el contraste del mismo.

- Display en estado



El display tras apenas ser encendido se encuentra inicialmente en este estado. Si no modifica dicho estado, esto indica una falla en la Inicialización.
Es necesario entonces verificar la alimentación del display o algún posible problema en el conexionado.

- Display en estado



Este mensaje indica el correcto funcionamientodel display, tras la inicialización del mismo.

CAPÍTULO 6

MEDICIONES

Capítulo 6

MEDICIONES

MEDICIONES INICIALES

Se realizaron mediciones por diferentes vías:

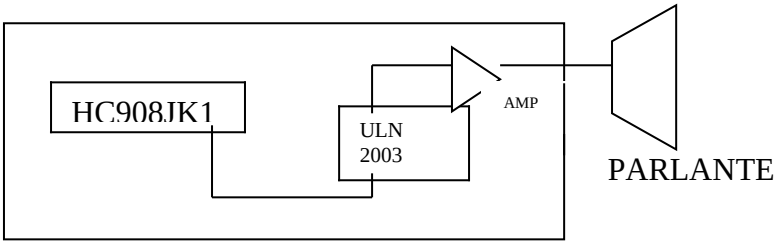
La medición fundamental en el transmisor es la de la frecuencia del tono generado. Ésta puede realizarse en principio mediante computadora, sin tener que recurrir a un frecuencímetro, osciloscopio u otro instrumento que pueda resultar poco accesible. Para lograrlo, se emplea la placa de audio, ya sea enviando la señal de salida por la entrada de línea u otro medio, y grabándola, para luego realizar un análisis de frecuencia de la onda grabada. Esta posibilidad de medir sin necesidad de instrumentos particulares se debe gracias a que dichas frecuencias se encuentran bastante por debajo de la fmuestreo/2 de la placa de audio, que es típicamente de unos 44,1KHz./2 = 22,05KHz. Existen sin embargo limitaciones tales como deformaciones que sufre la señal en el proceso de medición. Por ejemplo, si se utiliza un parlante, la señal cuadrada sufre cierta distorsión en sus armónicas. (Esquema A)

En el caso de la prueba para este sistema se utilizaron Matlab® y un software de instrumentación virtual que aprovecha la tarjeta de sonido del equipo.

Para el uso de Matlab, se desarrolló un sencillo programa de cinco líneas, donde se grafica la respuesta en frecuencia de la señal almacenada en un archivo de audio (ver Figuras 2 y 3), que puede analizarse con mayor detalle ampliando la escala como se ve en la fig.5.

El software de instrumentación virtual posee una interfaz gráfica como la mostrada en la fig.1 y actúa automáticamente al ejecutar el archivo de audio donde se ha guardado la señal.

Si bien existen limitaciones en su exactitud así como diferencias entre ambas mediciones, éstas resultan apropiadas para un primer análisis, dada la sencillez en su aplicación.



Esquema A. Modelo para la medición de tonos de audio.
En este caso es necesario recurrir a un driver (ULN 2003)
para manejar altos valores de corriente a la salida.

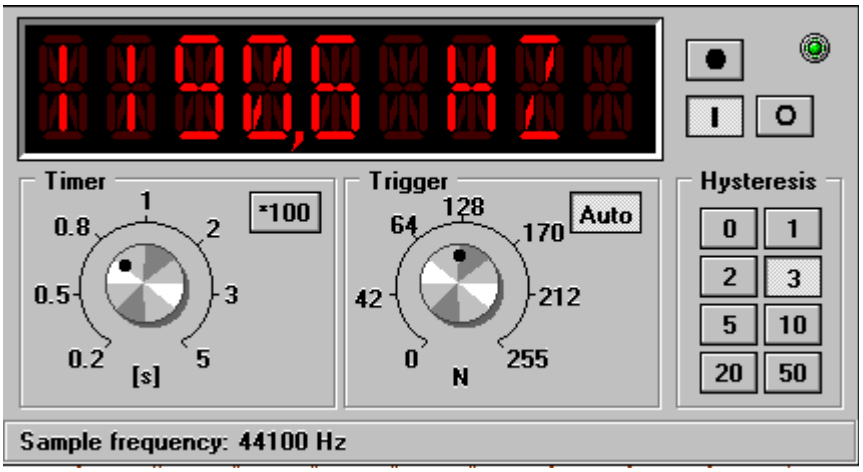


Figura1 . Interfaz del Frecuencímetro para la medición del tono Correspondiente al dígito 2

PROGRAMA CÁLCULO RESPUESTA EN FRECUENCIA

```
[x,Fs,nbits]=wavread ('c:\cc\tono2.WAV');
y=abs(fft(x,4096));
w=0:1:4095;
w=w./4096.*Fs;
plot (w,y);
```

Figura 2. Programa para el análisis espectral de la señal almacenada en el archivo tono2.WAV.

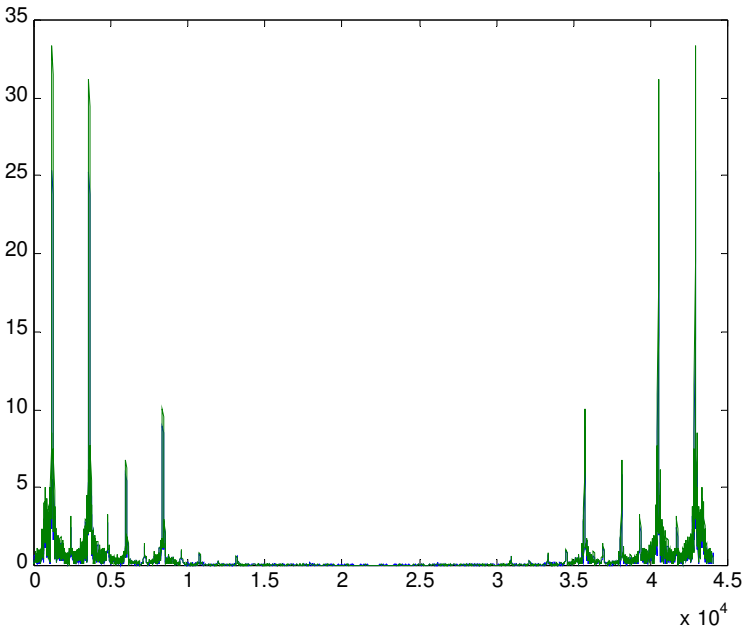


Figura 3. Gráfica de la respuesta espectral (tono 2.wav)

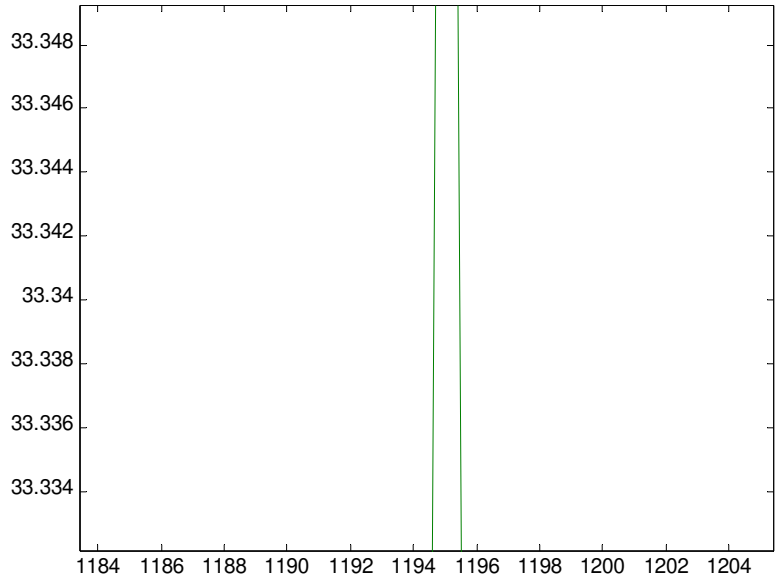


Figura 4. Zoom del gráfico de la figura 3

EMULACIÓN DE LA COMUNICACIÓN REAL

Existen dos posibilidades para efectuar una prueba emulando las características reales en la comunicación. Una comunicación de audio vía aérea con dispositivos que actúan como transmisor/receptor , o bien emular dichas condiciones reales típicas circuitalmente , sin necesidad de emplear dispositivos adicionales.

La primera alternativa es en principio ideal para obtener una respuesta más similar a la que tendría en funcionamiento en la realidad. Sin embargo, puede resultae menos viable debido al uso de dispositivos más específicos.

La segunda opción da una idea más bien esbozada o aproximada del comportamiento del sistema en condiciones reales, pero resulta más simple para implementar.

Para realizar una prueba dentro de condiciones más o menos reales, a la distorsión de la señal de salida del transmisor se le debe sumar ruido blanco gaussiano, típico en un sistema de comunicaciones FM / VHF. Para ello, se implementó un sumador con un ancho de banda lo suficientemente mayor al de transmisión como para considerar al ruido agregado como si fuera efectivamente blanco (cerca de 100KHz., esto es alrededor de 30 veces el de transmisión).

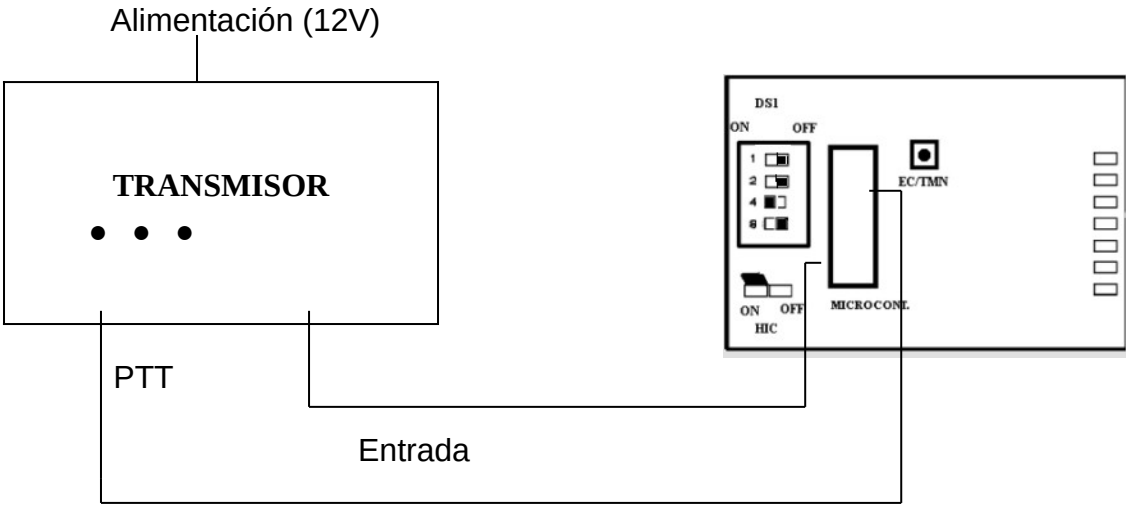
En principio, se optó por la segunda alternativa, a partir de la cual, se realizaron numerosas mediciones, a fin de obtener la respuesta más similar a la deseada, tanto en la generación de las condiciones reales como en la recuperación de la señal original en el transmisor. Puede observarse el sistema construido en la fig.4.

Surgieron sin embargo en esta etapa varios inconvenientes, principalmente en la simulación de senoidal más ruido blanco gaussiano,

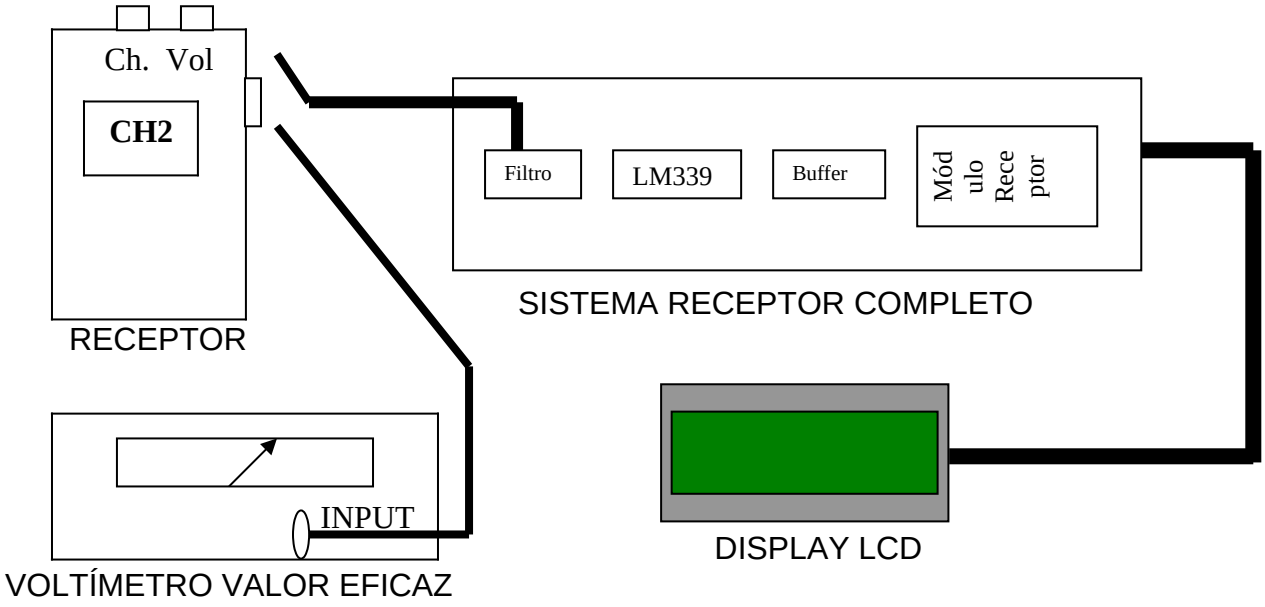
Tras estos problemas experimentales, se decidió que lo más recomendable era realizar una prueba en condiciones reales, con un trasmisor y un receptor portátil. Al transmisor se le

realizó un recorte en la etapa de potencia para lograr un empeoramiento de la relación señal a ruido, en tanto que al receptor se le quitó la antena, todo esto para poder simular condiciones reales en un entorno reducido.

A continuación se muestra el banco de medición experimental:



BANCO DE MEDICIÓN DEL TRANSMISOR



BANCO DE MEDICIÓN DEL RECEPTOR

Finalmente, concluimos con las mediciones relativas al ruido, en nuestro caso, medición del ruido a la entrada del receptor:

MEDICIÓN DEL SILENCIAMIENTO

Entre todas las posibles mediciones en un receptor, era conveniente –y prácticamente necesario- elegir una medición que fuera de gran utilidad práctica, y al mismo tiempo, evitara una dependencia con el receptor de VHF/UHF empleado, del cual toma la señal de entrada.
Por ello, se optó finalmente por la medición del mínimo nivel de silenciamiento al que responde: El nivel de silenciamiento en el que trabaja el sistema construido es independiente de cualquier receptor usado; siempre que se cumplan las condiciones de nivel mínimo, el sistema ha de responder. Esto se debe en la práctica, a que el funcionamiento de los receptores se ve limitado por un umbral de 20 dB, debajo del cual, el ruido "destruye" la señal transmitida. De modo que para una sensibilidad de silenciamiento de ese orden , se habrá garantizado el correcto funcionamiento del equipo sea el receptor que fuere al que se encuentre conectado.

A continuación se enumeran los pasos seguidos para esta medición:

Tras verificar el correcto funcionamiento del equipo (comunicación exitosa), en esas condiciones:

1. Conectar un voltímetro de audio sobre los terminales del parlante
2. Girar la perilla SQUELCH totalmente anti-reloj (obviar en el caso que el squelch esté abierto o desconectado, como es este caso).
3. Encender el equipo y ajustar la perilla VOLUMEN hasta obtener un valor apreciable en la escala
4. Registrar el nivel de ruido medido (empleando el voltímetro de valor eficaz)
5. Encender el PTT del transmisor sin entrada de modulación (sólo portadora)
6. Registrar el nuevo nivel de ruido medido (empleando el voltímetro de valor eficaz) y calcular el valor de silenciamiento obtenido según (1)..
7. Medir nuevamente bajo un nivel menor de silenciamiento

(1) $\text{Silenciamiento} = 20 \log \frac{\text{Valor Medido en paso 4.}}{\text{Valor Medido en paso 6}}$

Experimentalmente se obtuvo un nivel de **20 dB**, con lo que el sistema responde en el umbral de silenciamiento y es apto para usarse sobre cualquier sistema de enlace VHF/UHF.

CAPÍTULO 7
CONCLUSIONES

CONCLUSIONES

El resultado final del ensayo del equipo construido fue muy satisfactorio. El sistema cumplió con todos los objetivos de prestación y funcionamiento propuestos en la etapa de diseño.

En términos más específicos, el diseño del programa fue modificado sucesivamente en base a numerosas revisiones de los objetivos. Se partió de una meta inicial de comunicación exitosa a través de un enlace directo por línea, hasta el ensayo en condiciones reales de ambiente y con una baja relación señal-ruido, efectuándose una comunicación exitosa en el umbral de los 20 dB de silenciamiento. Con esto, el sistema es empleable en la práctica sobre cualquier sistema de enlace de VHF/UHF.

A esto, se le agregaron varias opciones extras, tales como distintos modos de funcionamiento del equipo: modo normal y modo emergencia en el caso del equipo transmisor y modo normal y modo freezing, en el caso del receptor. Asimismo se desarrollaron varias modalidades para la configuración del código de identificación, a saber, mediante un cuádruple dip-switch para el ingreso de dígito por dígito; mediante el uso de conversores A/D propios del microcontrolador, así como la posibilidad de usar un código predefinido cargado en memoria. Finalmente, posee un sistema de detección veloz en tiempo real, que se basa en un protocolo CCIR optimizado en velocidad.

La sencillez en su instalación y manejo puede apreciarse por ejemplo, en la conexión directa del display LCD con el módulo del microcontrolador , así como en el manual de operación.

La mínima complejidad circuital queda reflejada en los pocos elementos adicionales empleados y en las simplificaciones desarrolladas por software, tales como antirrebotes para los pulsadores o filtros desarrollados por software.

El costo final del sistema fue considerablemente bajo (menos de U$S 10, para el equipo transmisor y alrededor de U$S 20 para la plaqueta de la central), más aún en comparación con los costos de sistemas similares existentes en el mercado nacional, que superan los U$S 50 por plaqueta.

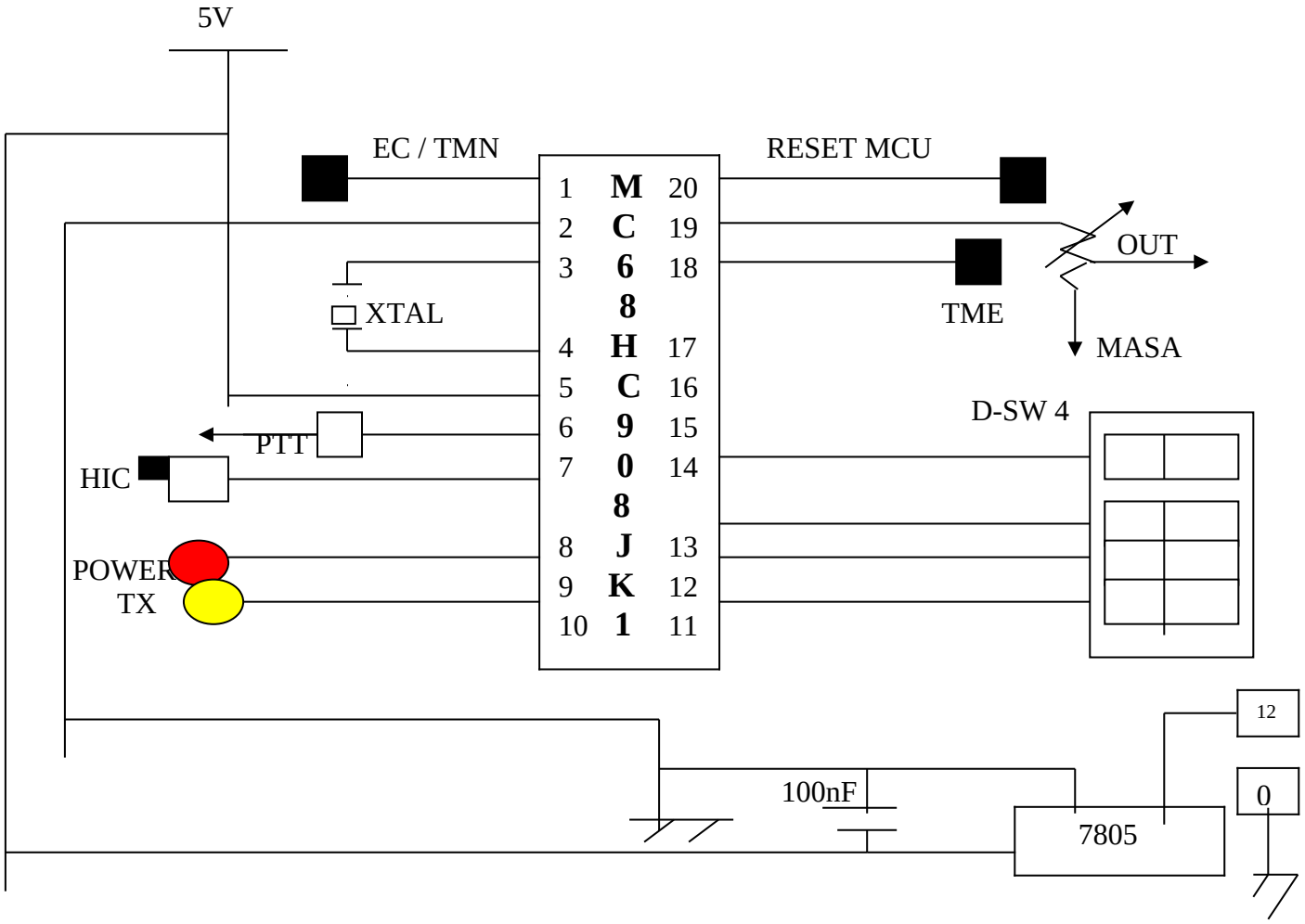
La flexibilidad en el diseño permite realizar modificaciones, tales como el número de dígitos por código o el formato de los tonos dado por el protocolo, con cambios mínimos en el programa y sin necesidad de ajustes en el hardware.

Por otro lado, la implementación real del diseño le sirvió a los autores para adquirir experiencia sobre problemas prácticos como por ejemplo los ruidos y otro fenómenos no deseados que se presentan en el proceso de decodificación de señales.

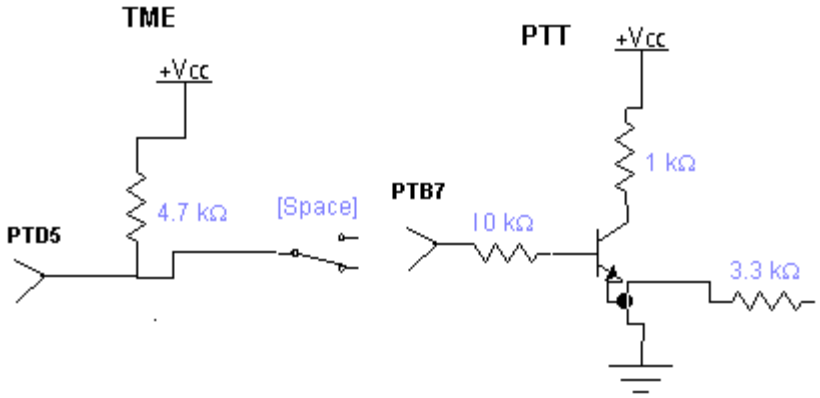
En relación a las posibles modificaciones y ampliaciones de las prestaciones del sistema, se podrían agregar más opciones tales como una entrada de micrófono en el transmisor para situaciones de emergencia, incluir mayor información en la comunicación como por ejemplo el barrio en el que se encuentra el móvil, destino y duración estimada del viaje, etc.,así como una interfaz con computadora en el caso de mayor complejidad en la comunicación. Finalmente, y con pocas modificaciones este sistema se podría adaptar para comunicaciones con una central de alarma.

DIAGRAMA ESQUEMÁTICO

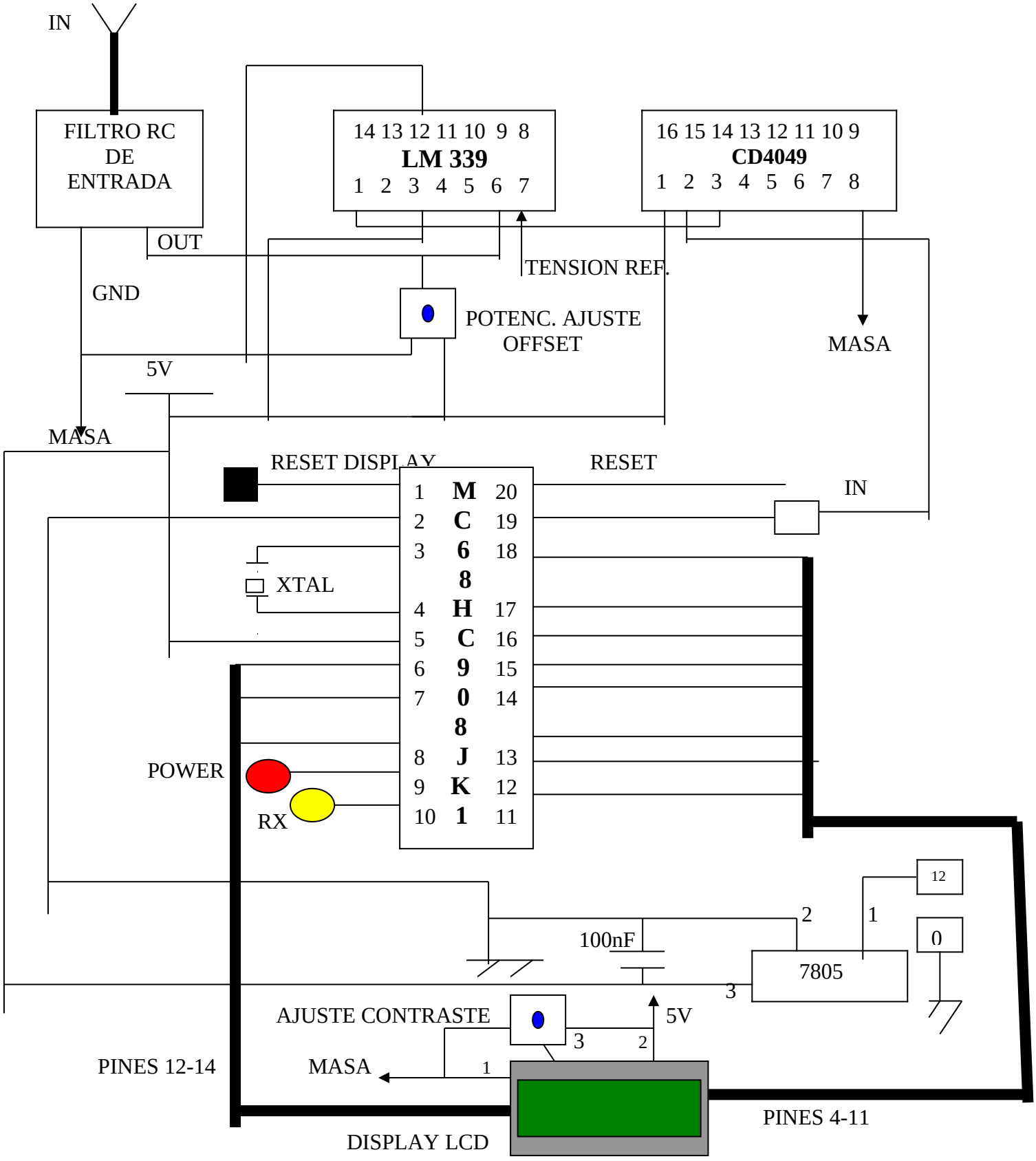
TRANSMISOR



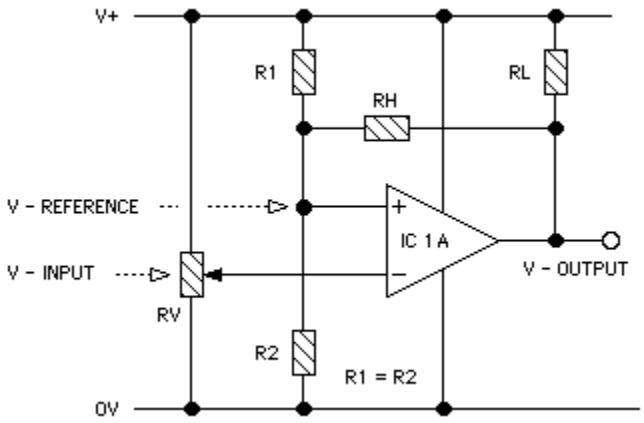
CIRCUITOS EQUIVALENTES



RECEPTOR



CIRCUITO EQUIVALENTE DEL COMPARADOR



RL = 1K
R1 = 7K
R2 = 1K
Rv = 0-50K
RH = 100K

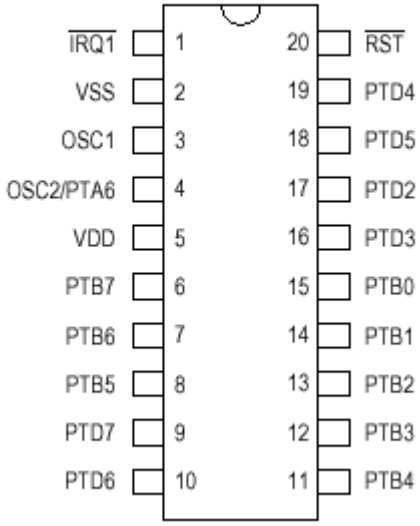
INTERFASE CON EL DISPLAY LCD

La interfase del microcontrolador receptor con el display LCD se realiza directamente mediante cable plano, siguiendo el interconexionado siguiente:

# PIN DISPLAY LCD	PIN MICROCONTROLADOR/ SALIDA
1	GND (MASA)
2	VCC (ALIMENTACIÓN 5V)
3	POTENCIÓMENTRO 10KΩ
4	PIN 3 PUERTO D
5	PIN 2 PUERTO D
6	PIN 5 PUERTO D
7	PIN 0 PUERTO B
8	PIN 1 PUERTO B
9	PIN 2 PUERTO B
10	PIN 3 PUERTO B
11	PIN 4 PUERTO B
12	PIN 5 PUERTO B
13	PIN 6 PUERTO B
14	PIN 7 PUERTO B

En tanto que también es posible realizar una conexión adicional para poder utilizar la opción de backlight que ofrece el display empleado.

MICROCONTROLADOR MC68HC908JK1/ MC68HC908JK3


IRQ1 1
VSS 2
OSC1 3
OSC2/PTA6 4
VDD 5
PTB7 6
PTB6 7
PTB5 8
PTD7 9
PTD6 10
20 RST
19 PTD4
18 PTD5
17 PTD2
16 PTD3
15 PTB0
14 PTB1
13 PTB2
12 PTB3
11 PTB4

APÉNDICE A
SOFTWARE DE DESARROLLO

APÉNDICE A

KIT Y SOFTWARE DE DESARROLLO

Sistema de desarrollo para el microcontrolador

El proyecto que realizaremos consiste en un sistema de desarrollo que soporta os microcontroladores **JK3** y el **MC68HC908JK1** (**JK1**) de la familia **08** de **MOTOROLA**. En la figura 1 se observa la conexión básica con la computadora.

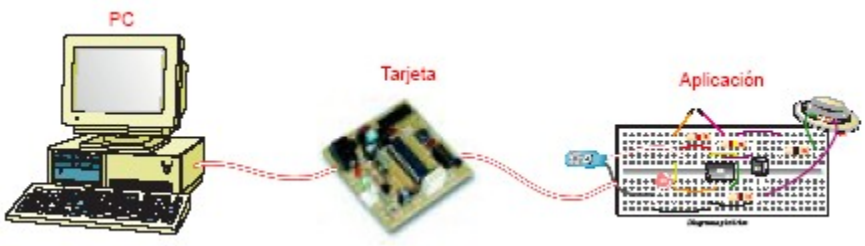
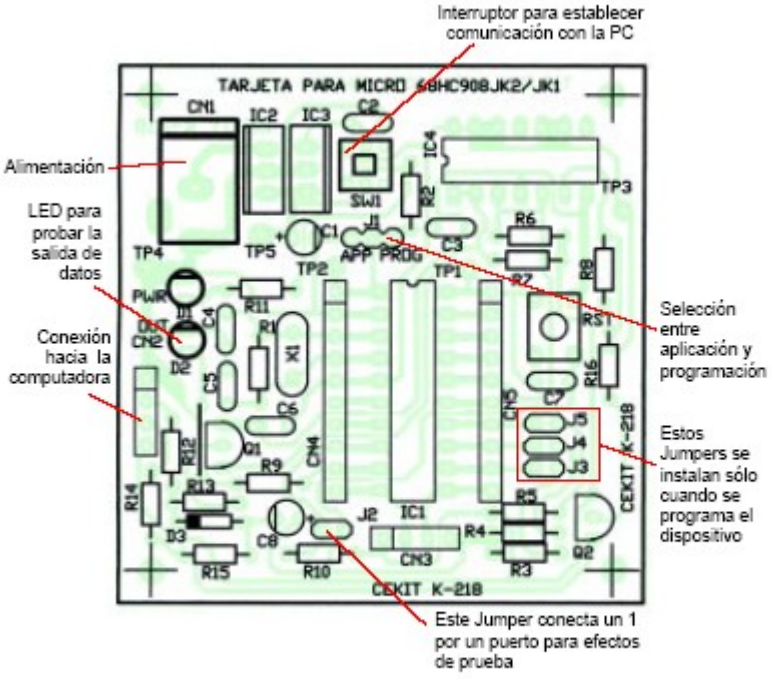


Figura 1. Conexión básica de la tarjeta con la computadora y con la aplicación

El sistema consta de un software y de una tarjeta en la cual se inserta el microcontrolador con los elementos básicos para su funcionamiento; la guía de montaje de esta tarjeta se puede apreciar en la figura 2. Posee entrada para alimentación de DC, conversores de nivel TTL a RS-232 que permiten conexión serial a una computadora para simulación y programación, circuito oscilador basado en cristal y LED rojo indicador de alimentación.



El jumper J1 selecciona el modo de funcionamiento. En la posición «PROG» el sistema arranca en modo SIMULACION-PROGRAMACION, permitiendo mediante el software instalado, realizar simulación y posterior programación del chip. J1 en la posición «APP»

permite que el sistema arranque en modo aplicación; es decir, ejecuta el programa grabado por el usuario.

SW1 es un suiche de dos polos y dos posiciones, que permite remover la alimentación completa del sistema cuando el software del PC así lo solicite para validar el código de seguridad interno del micro. El pulsador RST está conectado a la señal de reset del micro permitiendo el control por el usuario de esa función.

J3, J4 y J5 corresponden a tres de los pines del microcontrolador (PTB3, PTB2 y PTB1); para poder entrar al modo simulación-programación, ellos deben tener colocado el jumper respectivo. Si el usuario utiliza estos pines en su aplicación, deberá remover los jumpers cuando arranque el sistema en modo aplicación (J1 en posición APP).

La tarjeta posee un conector en línea macho de 3 pines (CN2), el cual permite la conexión directa al PC (para simulación-programación) y mediante los conectores en línea CN3, CN4 y CN5 se tiene acceso a todos los pines del chip, con los cuales podemos conectar el sistema de desarrollo a nuestra aplicación.

Adicionalmente, el sistema cuenta con un LED verde conectado a uno de los pines del microcontrolador (PD7) y un jumper que conmuta entre +VDD y tierra conectado al pin PB5; mediante estos dos elementos ilustraremos el manejo de salidas y entradas.

Ensamble

Con la guía de ensamble de la figura 2 se realizó el montaje de los componentes en el impreso de referencia K-218. se soldó primero los componentes de bajo nivel, tales como resistencias y condensadores; después se continua con los transistores y demás elementos. Al finalizar, se removió los excesos del fundente de la soldadura con un limpiador electrónico de contactos. Se realizó el ensamble del cable de interfaz con la computadora, haciendo los puentes como se ilustra en la figura 3 en el conector DB9: el pin 4 unido con el pin 6 y el pin 7 unido con el pin 8).En la figura 4 se muestra el aspecto físico de la tarjeta.

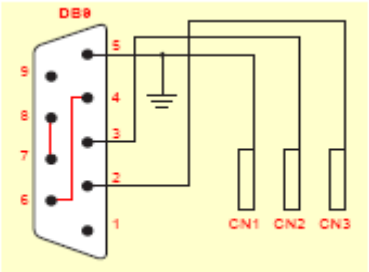


Figura 3 Cable que se conecta al puerto serial del PC para comunicarse con la tarjeta.

Figura 4. Aspecto final de la tarjeta ensamblada

Pruebas iniciales de la tarjeta

Para hacer las primeras pruebas a la tarjeta, se siguió el siguiente procedimiento.

1. Sin el microcontrolador JK3, se alimentó el circuito con un adaptador de 12VCD con terminación redonda.

2. Se realizó la medida de voltaje entre **TP4** (GND) y **TP3** (5V), debe ser alrededor de 5 VDC; se midió el voltaje entre **TP4** (GND) y **TP2** (IRQ); debe ser alrededor de 8 VDC;

3. Se removió la alimentación de la tarjeta y se colocó el micro **JK3** en la base.

Software de desarrollo WinIDE

El software mediante el cual se puede realizar edición, ensamblado, simulación ,programación y *debug* es completamente gratuito Para manipular este sistema, tenga en cuenta el siguiente procedimiento:

1. Instale el software **ICSJL** de pemicro.

2. Para verificar el funcionamiento de la tarjeta con el software **ICSJL**, conecte circuito al puerto serial de la computadora (DB9) y ejecute el programa **ICS08JLZ Development Kit**> **WinIDE Development Environment**. Allí se accede el segundo ícono de izquierda a derecha (**In-Circuit Simulator (EXE1)**),figura 1

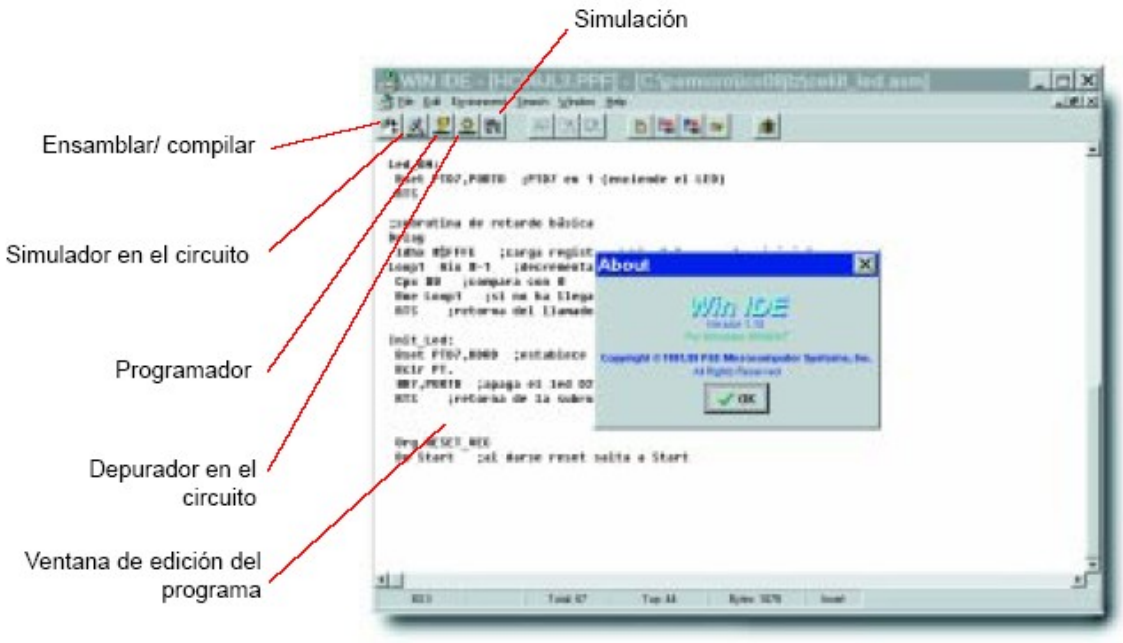


Figura 1.. WINIDE es el programa que permite la edición de los programas y además integrar en un solo entorno los procesos de ensamblaje o compilación, programación y depuración

3. Elija el puerto por el cual fue conectada la tarjeta y 9600 Baud como velocidad de comunicación, figura 2

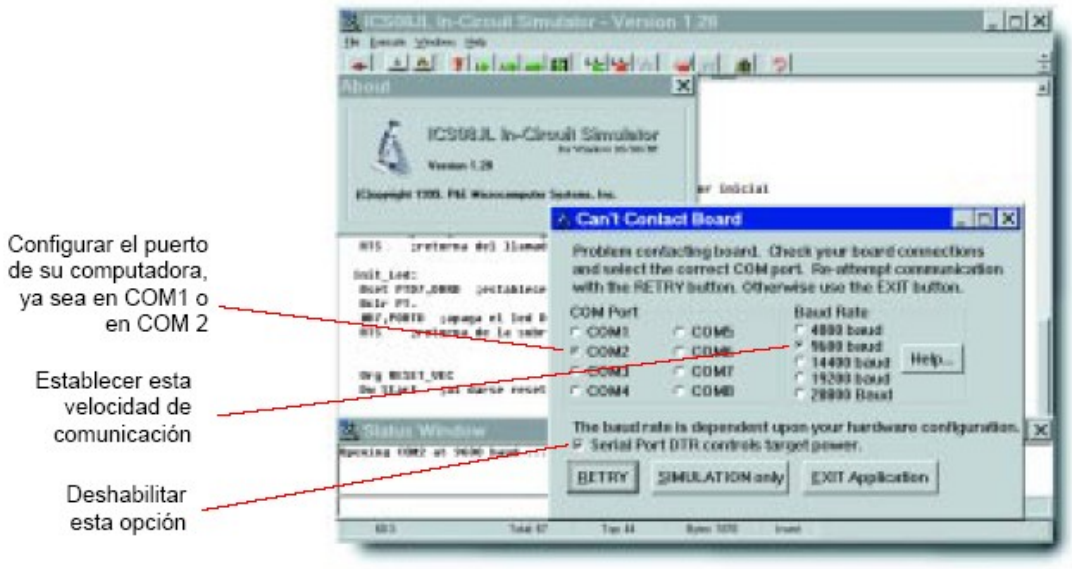


Figura 2. Para entrar al simulador se debe configurar el puerto serial a una velocidad de 9600 Baudios y deshabilitar la opción " Serial Port DTR controls target power"

4. Remover la selección del *Checkbox* "**Serial Port DTR controls target power**", para que el software le indique cuándo encender y apagar el circuito (esto es necesario para validar el código de seguridad, en caso que el MCU esté programado previamente).

5. Presionar el botón **RETRY**, para intentar comunicación con el circuito, si no obtiene comunicación luego de varios intentos de encendido y apagado, revise el circuito cuidadosamente y repita el proceso.

6. Al obtener comunicación con el circuito, debe aparecer en la pantalla una nueva interface, en la que se tienen ventanas de puertos, zonas de memoria y programa en *assembler*, el cual puede ejecutarse paso por paso en la tarjeta, figura 3

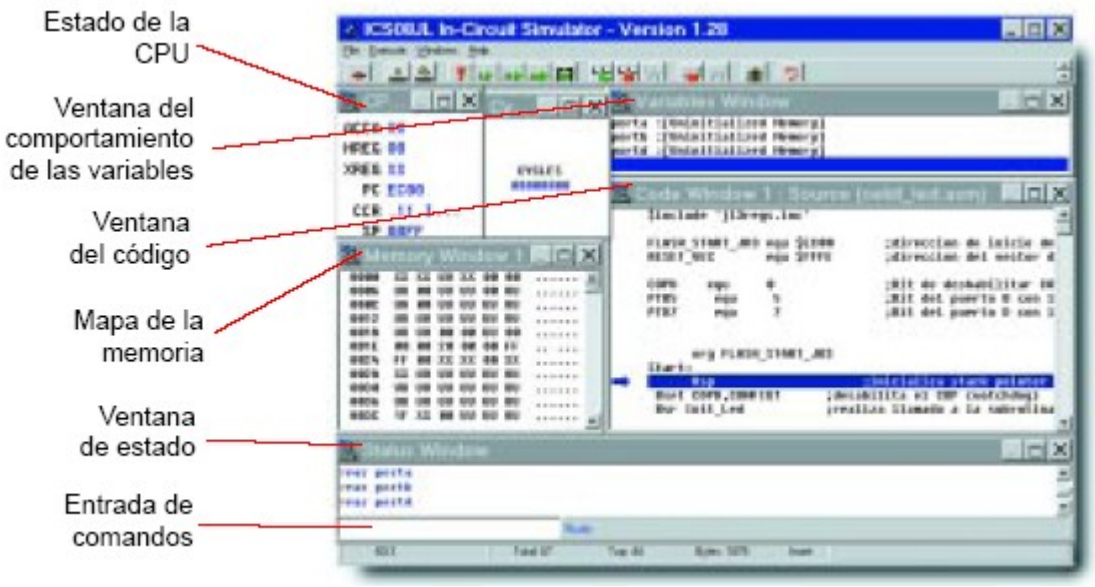


Figura 3. Pantallazo del simulador ICS08SL. Esta aplicación permite simular los programas en el circuito.

7. Para ilustrar la evaluación del hardware con la ayuda del software **ICSJLZ**, l circuito provee una salida (LED **D2**) y una entrada (Jumper **J2**). Coloque y remueva el Jumper **J2** de la tarjeta. Este cambio debe verse reflejado en la zona de puertos del software (esta es la forma como se pueden evaluar que las entradas de su proyecto funcionan adecuadamente). Para evaluar la salida acceda la línea de comandos del software (inferior izquierda) y modifique el estado del puerto D del MCU pin PD7; para esto se debe colocar el pin como salida colocando en el registro DDRD del MCU el valor $80 (hexadecimal), (comando: **DDRD 80**) y a continuación, el valor del pin en el registro PORTD del MCU en el estado requerido (para nuestro caso en "1" lógico), (comando: **PORTD 80**). Después de la ejecución de estos dos comandos, el LED **D2** debe encenderse, indicando la conexión adecuada (de manera similar se pueden evaluar las demás salidas del proyecto en particular).

8. La simulación de programas puede hacerse sin la conexión del circuito, accediendo el botón **SIMULATION only** (esta simulación es recomendable para zonas de programa que no tengan interacción directa con el hardware).

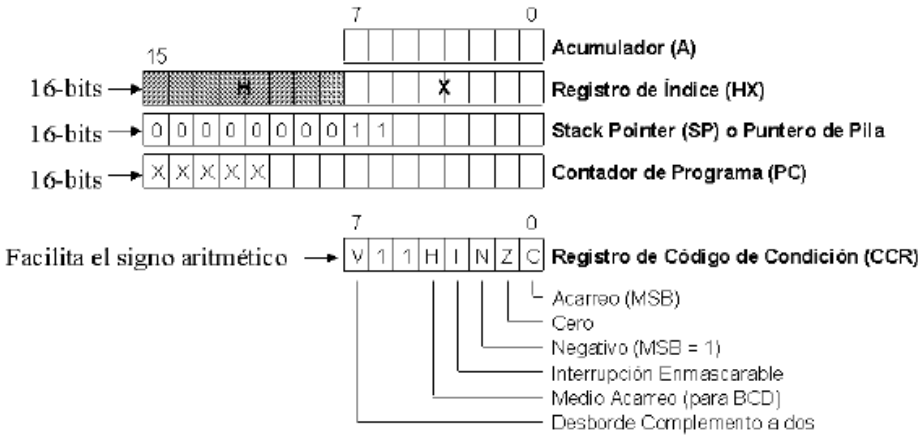
APÉNDICE B
MICROCONTROLADOR MC68HC908JK1/JK3

Apéndice B

Microcontrolador MC68HC908JK1/JK3

Arquitectura de la CPU MC68HC908JK1/JK3

Cada MC68HC908JK1/JK3 implementa este modelo de CPU sin tener en cuenta el tamaño o el conjunto de características propias. A continuación se verán las mejoras específicas que se incluyen en la arquitectura MC68HC908JK1/JK3



- El *registro de índice* se ha ampliado a 16-bits, ayudando a manejar mejor las tablas o estructuras de datos que son mayores de 256 bytes.
- El *puntero de pila* y el *contador de programa* son registros de 16-bits, sin tener en cuenta la memoria interna disponible. Más adelante se verá la pila (‘stack’).
- En *Power-On Reset*. Durante el Reset, los 8 bits más altos del registro de índice HX, de los 16 bits que tiene, se ponen a cero y el puntero de pila se inicializa a $00FF. Considerando que la mayoría de 68HC08 tienen mayor cantidad de RAM disponible, es probable que el usuario relocalizará el puntero de pila (‘stack pointer’). Sin embargo, esta característica ayuda a mantener la compatibilidad operacional con el software existente del 68HC05.
- En la CPU del MC68HC908JK1/JK3 el *bit V*, del registro de código de condición (CCR) facilita los cálculos aritméticos con signo. Esta mejora permite a los programadores de lenguaje ensamblador y a los compiladores, realizar cálculos de direccionamiento mucho mejor.

Descripción de los pines del MC68HC908JK1/JK3

En la siguiente figura se puede observar la distribución de pines del microcontrolador.
Y en la tabla la función que desempeña cada uno de ellos

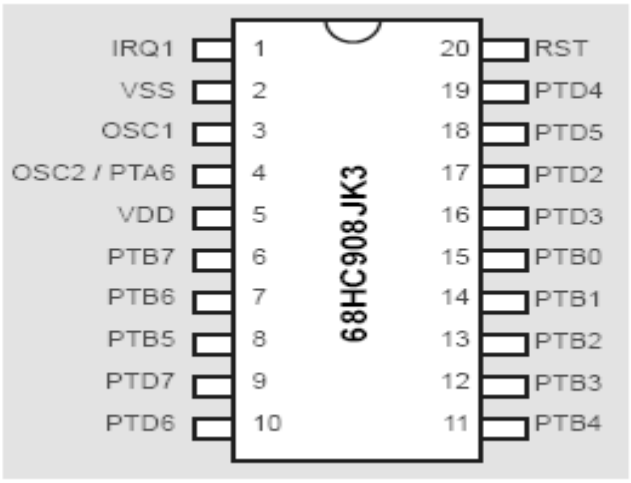


Diagrama de pines del integrado

Nombre del pin	Descripcion	Entrada/salida	Nivel de voltaje
VDD	Alimentación positiva	Entrada	5v ó 3v
VSS	Tierra	Salida	0v
RST	Reset activo en bajo con resistencia interna de pull-up	Entrada	VDD
IRQ1	• Interrupcion externa • Posee resistencia interna de pull - up • Pin usado para seleccionar el modo de arranque (programación o aplicación)	Entrada	VDD
OSC1	Entrada del oscilador	Entrada	Análogo
OSC2	Salida del oscilador	Salida	Análogo
PTB (0:7)	• Puerto I/O de 8 bits	Entrada/salida	VDD
	• 8 entradas análogas, ADC(0:7)	Entrada	Análogo
PTD (2:7)	• Puerto I/O de 6 bits	Entrada/salida	VDD
	• PTD (3:2) 2 canales de ADC, ADC(8:9)	Entrada	Análogo
	• PTD (4:5) 2 pines de temporizador	Entrada/salida	VDD
	• PTD (6:7) salida en colector abierto de máximo 25 mA	Entrada/salida	VDD

Descripción de las funciones de cada uno de los pines

Tiempos de los Ciclos Internos de la CPU MC68HC908JK1/JK3

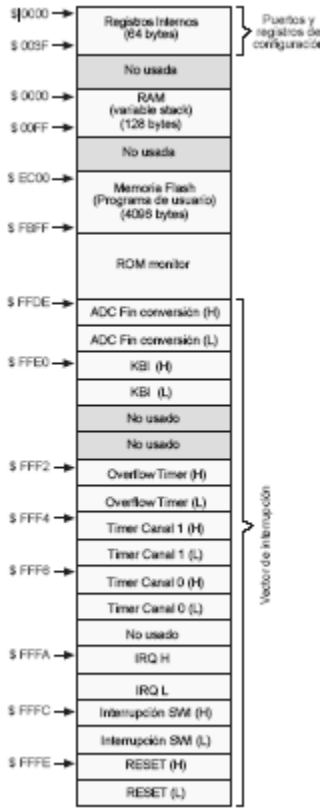
El MC68HC908JK1/JK3 utiliza cuatro fases del reloj interno en cada ciclo de ejecución de la CPU. Si el MC68HC908JK1/JK3 está gobernado por un cristal, el ciclo de ejecución es un cuarto de la frecuencia del cristal. A este ciclo se le llama *ciclo de bus* o *ciclo de instrucción*.

En el MC68HC908JK1/JK3 , todos los tiempos de cada instrucción se especifican en *ciclos de bus*. Por ejemplo, un reloj de entrada de 32 MHz producirá una frecuencia de bus de 8 MHz. Un *ciclo de bus* de una instrucción se ejecutará en 125 ns o 1 dividido por 8 MHz.

La mayoría de 68HC08 incluyen un circuito PLL interno que a partir de una frecuencia más baja de cristal, genera internamente una frecuencia de reloj mucho más alta que la del cristal. Para más información sobre la generación del reloj y el circuito PLL, se puede ver en el módulo de Generación de Reloj (CGM) de esta guía didáctica.

Mapa de memoria interna del microcontrolador

Allí se encuentran las direcciones de los registros internos, la posición que ocupa la RAM y la parte asignada al programa del usuario. En la zona alta de memoria se encuentran los vectores de interrupción y en ellos se encuentrala dirección (2 bytes) con la cual se carga el contador de programa (PC) cuando se presenta alguna de las interrupciones



Juego de Instrucciones del MC68HC908JK1/JK3

El MC68HC908JK1/JK3 añade 28 instrucciones al juego de instrucciones del 68HC05, remarcadas en la tabla siguiente. Muchas de estas instrucciones soportan el registro del índice extendido a 16 bits y el puntero de pila (‘stack pointer’) es totalmente relocalizable.

Movimiento de Datos	LDA, LDX, STA, STX, TAX, TXA, **LDHX, MOV, PSHA, PSHH, PSHX, PULA, PULH, PULX, STHX**
Aritméticas	ADD, ADC, SUB, SUBC, MUL, **DAA, DIV**
Manipulación de Datos (Inc/Dec/Neg/Clr)	INCA, INCX, INC, DECA, DECX, DEC, CLR,NEGA, NEGX, NEG, **AIS, AIX, CLRH**
Manipulación de Datos (Rotación/Desplazamiento)	ROLA, ROLX, ROL, RORA, RORX, ROR, LSLA, LSLX, LSL,LSRA, LSRX, LSR, ASRA, ASRX, ASR
Manipulación de Datos	BSET, BCLR
Lógicas	AND, ORA, EOR, COMA, COMX, COM, **NSA**
Test de Datos	CMP, CPX, BIT, TSTA,TSTX, TST, BRCLR, BRSET, **CPHX**
Bifurcación	BRA, BRN, BSR, BHI, BLO, BHS, BLS, BPL, BMI, BEQ, BNE, BCC, BCS, BHC, BHCC, BHCS, BMC, BMS, BIL, BIH, **BGE, BGT, BLE, BLT, CBEQ, CBEQA, CBEQX, DBNZ**
Salto/Retorno	JMP, JSR, RTS
Control	SEC, CLC, SEI, CLI, SWI, RTI, RSP, NOP, WAIT, STOP, **TAP, TPA, TSX, TXS**

Se han añadido varias instrucciones de bifurcación condicional, como: el ajuste decimal del acumulador DAA y la instrucción NSA (Nibble Swap Accumulator). La instrucción de división DIV sin signo, dividiendo 16 por 8 ejecutándose en sólo 7 ciclos de bus. La instrucción multiplicación (MUL) se ha reforzado para que se ejecute en sólo 5 ciclos de bus, en lugar de los once ciclos necesarios para el 68HC05. Esta mejora del juego de instrucciones del, MC68HC908JK1/JK3 proporciona mayor eficacia de código y funcionamiento, recogiendo múltiples instrucciones y realizando la misma tarea con una sola instrucción. Más adelante se verán en detalle estas nuevas instrucciones.

Nuevos Modos de Direccionamiento del MC68HC908JK1/JK3

El MC68HC908JK1/JK3 tiene varios nuevos modos de direccionamiento y son: dos modos de direccionamiento indexado, dos modos de direccionamiento del puntero de pila y cuatro modos de direccionamiento de movimiento de memoria a memoria.

Modo de Bajo Consumo del MC68HC908JK1/JK3 (Wait y Stop)

Modo Wait:

Se *entra* ejecutando la instrucción WAIT (Consumo típico 50% de IDD en modo RUN)

Efectos:

- El reloj de la CPU se desactiva.

- El reloj del bus sigue activo.
- Los periféricos individuales se pueden desactivar para mejorar el consumo.

Se *sale* del modo 'Wait' por un Reset o por una interrupción externa o interna.

Modo Stop:

Se *entra* ejecutando la instrucción STOP (Consumo tipico: I_{DD} de 1µA a 3 µA)

Efectos:

- El reloj de la CPU se desactiva.
- El reloj del Bus opcionalmente se desactiva.

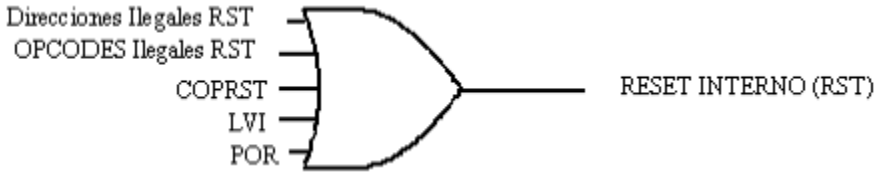
Se *sale* del modo 'Stop' por un Reset, por una interrupción externa o por una interrupción TBM (si el reloj del bus está activado).

Módulo del Reset y las Interrupciones

En esta parte se explica el proceso del Reset y de las Interrupciones en la MCU. Se describen las diferencias entre las Interrupciones y el Reset. Se identifican las diferentes fuentes de Interrupción y Reset. Se describe el proceso de restablecimiento después de un Reset. El Reset y las interrupciones son respuestas a los eventos excepcionales durante la ejecución de un programa.

Fuentes de Reset

- **Externas:** *Power on Reset (*POR) o con el *Pin de Reset puesto a masa*
- **Internas:** *COP* o *LVI* o *Opcodes Ilegales* o *Direcciones Ilegales*



El Reset puede ser causado por una señal en el pin de reset externo o por una señal de reset interna. Las señales de reset internas se pueden generar por el módulo COP o por el módulo LVI. Otras fuentes de reset internas son por un 'opcode' ilegal y por una dirección ilegal.
Un reset detiene la ejecución del programa en la MCU. Una vez que se ha procesado el reset, la MCU vuelve inmediatamente a las condiciones de inicio conocidos y empieza la ejecución del programa desde una posición de memoria definida por el usuario.

Proceso de un Reset

- Se genera un reset por una *dirección ilegal* cuando la CPU intenta sacar una instrucción de una dirección que no está definida en el mapa de memoria. Este tipo de reset

proporciona una protección adicional del sistema y devuelve a la CPU a un estado conocido cuando se usa una dirección inválida.

- Se genera un reset por un '*opcode ilegal*' cuando la CPU descifra una instrucción que no está definida en el juego de 'opcode'. Ruido excesivo o código que se intenta ejecutar incorrectamente desde el espacio de los datos, pueden causar este evento y pueden devolver a la CPU a un estado conocido.

- Se genera un reset por el módulo *COP*, cuando hay un desbordamiento del contador del COP, indicando que el timer del COP no se reparó a tiempo. Èsta es una protección del sistema contra software no correcto, que devolverá a la CPU a un estado conocido.

- Se puede generar opcionalmente un reset por módulo *LVI*, cuando V_{DD} cae por debajo de un valor del punto de disparo. Esta característica protege contra el funcionamiento incorrecto de la MCU durante condiciones de bajada de tensión ('brown-outs'). Para el reset LVI, la línea de reset permanece en estado bajo durante 4096 ciclos de reloj CGMXCLK, después de que V_{DD} se restablece, permitiendo asi estabilizar el reloj.

- Power-On-Reset (POR) es un reset interno causado por una transición positiva desde 0, en el pin $V_{DD.}$ Todos los relojes internos de la CPU y todos los módulos de la MCU se mantienen inactivos durante 4096 ciclos de reloj CGMXCLK, para así permitir estabilizar el oscilador. Durante este tiempo, el pin RST permanece en estado bajo. El reset POR solo se puede activar cuando V_{DD} cae a 0 voltios. El POR no es un detector de bajadas de tensión ('brown-out'), detector de bajo voltaje o detector de 'glitch'.

Registro del estado del reset SIM

El registro de estado del reset SRSR del SIM, contiene seis indicadores que muestran la fuente del último reset.

SRSR	Bit 7	6	5	4	3	2	1	Bit 0
Leer:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
Escribir:								
POR:	1	0	0	0	0	0	0	0

- Si el *bit* POR del Power-On-Reset se pone a 1, el último reset fue causado por el circuito POR.

- Si el *bit* PIN del pin de reset externo se pone a 1, el último reset fue causado por un dispositivo externo poniendo el pin de reset a un nivel bajo.

- Si el *bit* COP se pone a 1, el último reset fue causado por el contador del COP, que cuenta antes de que le fuera dado servicio.

- Si el *bit* ILOP de 'opcode' ilegal se pone a 1, el último reset fue causado por un 'opcode' ilegal.

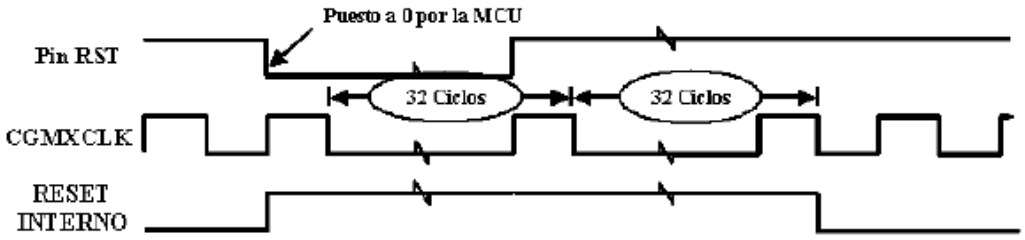
- Si el *bit* ILAD de dirección ilegal se pone a 1, el último reset fue causado por un ‘opcode’ recuperado de una dirección ilegal.

- Si el *bit* LVI de voltaje bajo se pone a 1, el último reset fue causado porque el LVI detectó un voltaje por debajo del punto de disparo seleccionado.

Los indicadores (‘flags’) en el registro de estado del reset del SIM se borran cuando se lee este registro. Si este registro de estado del reset del SIM no se ha leído después de varios resets, tendrá señalizados estos resets. Entonces este registro puede ayudar a depurar problemas de código.
En caso de un Reset en la aplicación, el registro SRSR ayuda a determinar qué acción tomar. La aplicación puede necesitar de una nueva reestructuración y de una inicialización por un reset del POR que no se necesita con un reset del COP. Otras fuentes de reset pueden necesitar diagnósticos o servicios únicos.

Tiempos del Reset Interno

Ahora que se han visto los diferentes tipos de reset interno, se pueden ver los tiempos internos del reset. Todos los tipos de reset interno, excepto el POR, ponen el pin RST a un nivel bajo durante 32 ciclos de reloj CGMXCLK, para permitir hacer un reset a los dispositivos externos.



La MCU se mantiene en reset durante 32 ciclos de reloj CGMXCLK después de que vuelva a un nivel alto el pin RST, para permitir estabilizar los dispositivos externos. El pin RST se prueba para verificar si el pin RST todavía está en estado bajo. En ese caso, indica que ha ocurrido un reset interno y el bit de reset interno se pone a 1, en el registro de estado del reset del SIM.

Tipos de Interrupción

- **Hardware**: Se inician por un pin hardware, que puede ser el pin IRQ o por los puertos de Entrada/Salida, por los puertos serie SCI/SPI o por el módulo TIM. Usan un vector de interrupción y una rutina de servicio. Se pueden enmascarar.

- **Software**: Se ejecutan como parte del flujo de instrucción. Se procesan como una interrupción hardware. No se puede enmascarar.

Hay dos tipos de interrupciones, *hardware* y *software*. Una interrupción no detiene la MCU o el funcionamiento de la instrucción que se está ejecutando, sino que sectoriza el contador de programa a una rutina de servicio de interrupción. Cuando ocurre un evento de interrupción, la

MCU primero completa la instrucción en curso y entonces cambia la secuencia de ejecución del programa para responder al evento.
Una interrupción es similar a un reset en que provoca a la MCU recuperar una nueva dirección para el contador del programa y pone a 1 el bit de máscara de interrupción (bit I). Al contrario del reset, las interrupciones solo suspenden temporalmente la ejecución normal del programa para que el procesador pueda dar servicio a la interrupción. Después de que se ha dado servicio a la interrupción, el procesador vuelve de donde salió para ejecutar el código normal del programa.

Proceso de una Interrupción

Se genera una interrupción hardware por una condición interna o externa del hardware y se puede iniciar por un pin hardware o por un módulo de la MCU. Cuando ocurre una interrupción hardware, el contexto del programa se apila (en el 'stack'), el bit I del registro de código de condición se pone a 1 y se recupera el vector de interrupción.
Se genera una *interrupción software* como resultado de la instrucción SWI. Una interrupción software siempre se ejecuta como parte del flujo de la instrucción. La diferencia importante entre las interrupciones software y hardware, es que las interrupciones software no se pueden enmascarar. Esto significa que el valor del bit I no tiene efecto en interrupciones software. Por otra parte, una interrupción software se procesa de la misma manera que una interrupción hardware.

Fuentes de Interrupción Hardware

El pin IRQ se puede usar para activar interrupciones de hardware externas. Dependiendo de la configuración de la MCU, una interrupción por IRQ se genera por un nivel bajo o por una transición de nivel alto a un nivel bajo en el pin IRQ. Este tipo de interrupción se puede usar para supervisar sistemas externos o eventos.
En la mayoría de las MCU del 68HC08, también se puede generar una interrupción usando pines adicionales del puerto I/O. Estos pins se les denominan KBI (interrupciones de teclado) y normalmente se usan de las entradas de una matriz de teclas. Estas entradas tienen 'pullups' programables que generan una interrupción cuando pasan a un nivel bajo.
Por ejemplo, una matriz de 16 teclas se organiza normalmente con 4 filas y 4 columnas. Las cuatro filas se pueden conectar a las entradas de KBI. Cuando se pulsa una tecla, la entrada de la fila correspondiente se pone a un nivel bajo y se genera una interrupción sin ninguna circuitería lógica adicional. La rutina de servicio de interrupción detecta los rebotes de la tecla y determina qué tecla fue pulsada examinando las columnas.
El 'timer' de 16-bits del Módulo TIM puede generar varias interrupciones diferentes que dependen del modelo en particular. El 'output compare', el 'input capture' y las funciones de desbordamiento del 'timer' pueden generar interrupciones. Algunos modelos también tienen una característica de interrupción en tiempo real. Se pueden usar estos tipos de interrupciones para procesar eventos basados en una referencia de tiempo.
Para MCUs que tienen un Módulo SCI o SPI, los puertos serie pueden generar una variedad de interrupciones que dependen del tipo. Las interrupciones SCI y SPI incluyen un registro de transmisión vacío y un registro de transmisión completa. Se pueden usar estos tipos de interrupciones para procesar eventos de comunicaciones de serie. Otros periféricos como el CAN y el USB también pueden generar interrupciones.

Fuentes de Interrupción

Las fuentes de interrupción usan una dirección del vector y una prioridad. La tabla resume las 16 diferentes fuentes de interrupción asociadas a la MCU del MC68HC908JK1/JK3. La arquitectura del 68HC08 puede manejar hasta 128 diferentes fuentes de interrupción y 'reset'.

FUENTE	Direcc. Vector	Flag	Máscara	INT	Prioridad
TimeBase	$FFDC - $FFDD	TBIF	TBIE	IF16	16
ADC Conv. Completa	$FFDE - $FFDF	COCO	AIEN	IF15	15
Pin teclado KBD	$FFE0 - $FFE1	KEYF	IMASKK	IF14	14
SCI Trans. Completa	$FFE2 - $FFE3	TC	TCIE	IF13	13
SCI Trans. Vacía		SCTE	SCTIE		
SCI Entrada desocupada	$FFE4 - $FFE5	IDLE	ILIE	IF12	12
SCI Recepción completa		SCRF	SCRIE		
SCI Recepción excedida	$FFE6 - $FFE7	OR	ORIE	IF11	11
SCI Flag de Ruido		NF	NEIE		
SCI Error de cuadro		PE	FEIE		
SCI Error de Prioridad		PE	PEIE		
SPI Transmisión Vacía	$FFE8 - $FFE9	SPTE	SPTIE	IF10	10
SPI Recepción completa	$FFEA - $FFEB	SPFR	SPRIE	IF9	9
SPI Desbordamiento		OVRF	ERRIE		
SPI Modo Fallo		MODF	ERRIE		
TIM2 Desbordamiento	$FFEC - $FFED	TOF	TOIE	IF8	8
TIM1 Canal 1	$FFEE - $FFEF	CH1F	CH1IE	IF7	7
TIM2 Canal 0	$FFF0 - $FFF1	CH0F	CH0IE	IF6	6
TIM 1 Desbordamiento	$FFF2 - $FFF3	TOF	TPOIE	IF5	5
TIM 1 Canal 1	$FFF4 - $FFF5	CH1F	CH1IE	IF4	4
TIM1 Canal 0	$FFF6 - $ FFF7	CH0F	CH0IE	IF3	3
PLL	$FFF8 - $FFF9	PLLF	PLLIE	IF2	2
IRQ	$FFFA - $FFFB	IRQF	IMASK1	IF1	1
SWI	$FFFC - $FFFD	No	No	No	0
Reset	$FFFD - $FFFF	No	No	No	0

Cada fuente de interrupción tiene su propia y única dirección de vector. Esto puede eliminar la necesidad de que el software consulte dentro de una rutina de servicio de interrupción para determinar la fuente correcta de interrupción y producir un servicio de interrupción más rápido. Cada fuente de interrupción también tiene su propio y único indicador en el registro de estado de interrupción que se puede consultar. Se puede desactivar una fuente de interrupción por un borrado del bit de la única fuente de interrupción.
Cada tipo de interrupción tiene una prioridad predefinida asociada. La interrupción del módulo TBM tiene la prioridad más baja y una IRQ externa tiene la prioridad más alta. Cuando ocurren múltiples interrupciones, la CPU primero mira la prioridad de los eventos y servicios del evento con la prioridad más alta, con los otros pendientes. Un evento de 'reset' tiene la prioridad por encima de todos los eventos de interrupción.
El bit I de máscara de interrupción global, activa o desactiva toda interrupción que se procesa con la excepción de interrupción software SWI. Los eventos de interrupción generados por módulos periféricos internos se pueden enmascarar y solo se reconocen si se pone a 0 el bit I. Estos periféricos tienen típicamente máscaras locales para activar o desactivar tipos específicos de interrupciones.
Por ejemplo, el Módulo TIM tiene una mascara de interrupción separada y el bit se activa por desbordamiento y por cada canal del 'timer'. Los 'resets' no se pueden enmascarar, pero algunos módulos internos se pueden desactivar para que no puedan generar un 'reset'. Los módulos COP y LVI son ejemplos de fuentes potenciales de reset que se pueden desactivar fuera de un 'reset'.

Contexto de Intercambio

Seguidamente se puede ver cómo el Módulo SIM usa esta información para abastecer las interrupciones.
Se denomina proceso de excepción el que determina qué tipo de gestión se necesita. El proceso de excepción a veces se maneja a través de tareas discretas llamadas ‘contexto que cambia’. El proceso de excepción es diferente para el reset y las interrupciones. Sin embargo, las tareas de proceso son las mismas.
Primero, el módulo SIM reconoce los eventos y realiza el arbitraje. Primero se procesa el evento de prioridad más alto. Antes de que se saque el vector, el bit I se pone a 1 para prevenir futuros eventos de interrupción y el contexto actual de la CPU se guarda en el ‘stack’. Finalmente se ejecuta la rutina de servicio de interrupción o el manejo de la excepción.

Reconocimiento del Reset y de las Interrupciones

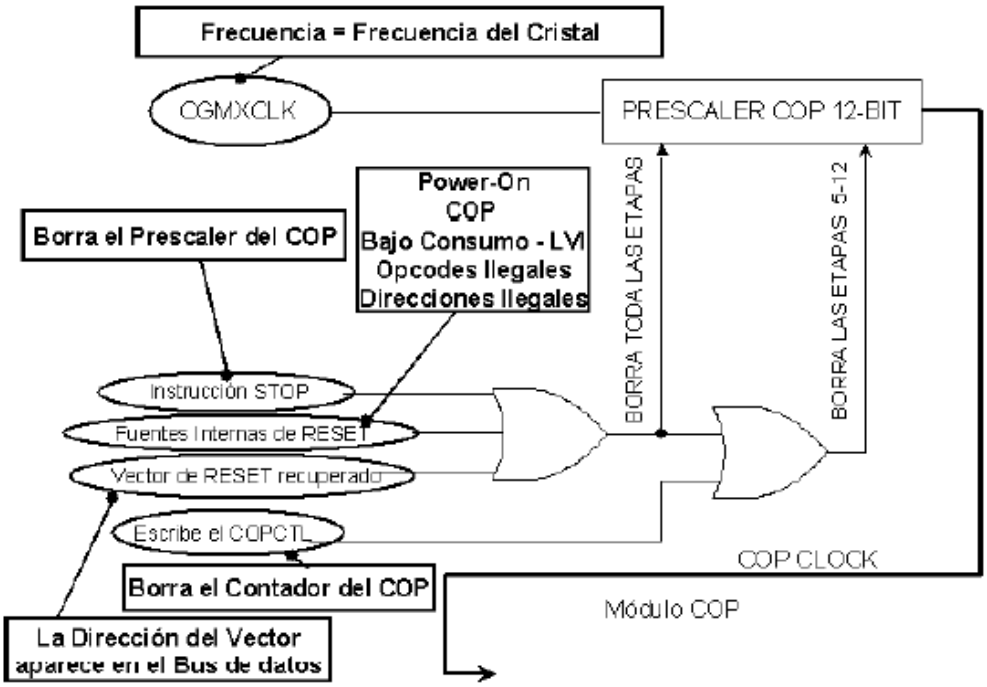
Durante la fase de reconocimiento, se reconocen los resets y se actúa inmediatamente. Las interrupciones se reconocen durante el último ciclo de ejecución de la instrucción. El tiempo de reconocimiento de la interrupción depende de cuando ocurre la interrupción. Si ocurre una interrupción del último ciclo de la instrucción en curso, se reconocerá durante el último ciclo y entonces se actuara. Si ocurre una interrupción durante el último ciclo de la instrucción en curso, no se reconocerá hasta el último ciclo de la siguiente instrucción.

Módulo COP (Computer Operating Properly)

En esta parte se describen las características y la configuración del modulo COP. El 'timer' COP, también conocido como 'Watchdog Timer' se basa en un contador que corre libremente y que se puede borrar por el código del usuario. El COP permite a la CPU recuperarse de eventos inesperados como el llamado 'runaway software' (ejecución incorrecta del software) y errores en el proceso de software.
Para usar el COP, simplemente hay que activar y seleccionar el periodo de 'timeout' deseado. El COP hace un 'reset' si no ha habido un Reset dentro del periodo de 'timeout'. El 'timeout' es el exceso de tiempo en la espera de una señal determinada. Lo más importante que siempre se debe recordar, es que hay que servirlo antes de que el periodo de interrupción expire. Si se activa el 'timer' del COP y no se le da servicio dentro del periodo de interrupción, el COP hará un reset a la CPU.
Seguidamente se verán las diferentes señales y bloques asociados al módulo COP, empezando con el prescaler del COP.

Prescaler del COP

Partiendo de la señal CGMXCLK, que es la señal de salida del oscilador a cristal, cuya frecuencia CGMXCLK es igual a la frecuencia del cristal, entra al prescaler de 12 bits y la señal de salida es el reloj COP.

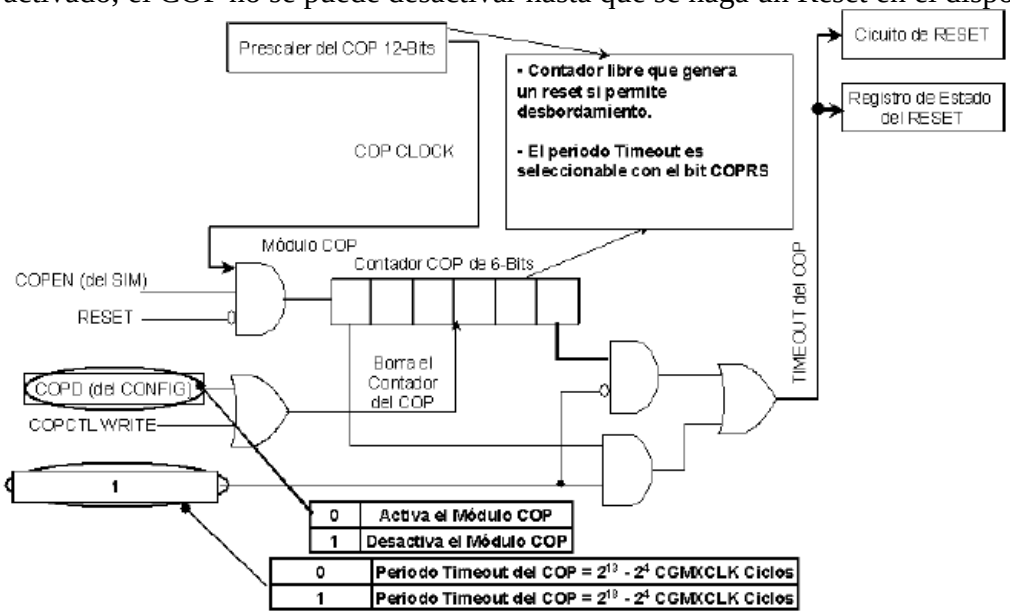


La instrucción STOP borra automáticamente el prescaler del COP. Esto se puede evitar, desconectando inadvertidamente el COP, con una instrucción STOP para desactivar la instrucción STOP. Cuando se desactiva la instrucción STOP y se ejecuta una instrucción STOP resulta un 'reset' por un 'opcode' ilegal.
Las fuentes de 'reset' internas por Power-On, por el módulo COP, por el módulo LVI, por un 'opcode' ilegal o por una dirección ilegal, borraran el prescaler y el contador del COP.

Ocurre la recuperacion de un vector de reset, cuando la dirección del vector aparece en el bus de datos. Un vector de reset recuperado borra el prescaler del COP.
Escribiendo algún valor en el registro de control COPCTL del COP, se borra el contador del COP y se borran los bits de las etapas 5 al 12 del prescaler.

Diagrama de Bloques del COP

La siguiente figura muestra las señales restantes del COP y sus bloques asociados. El COP se activa fuera del reset. Si no se desea el funcionamiento del COP se debe desactivar inmediatamente poniendo a 0 el bit COPD del registro de configuración del COP. El registro de configuración es un registro que se escribe una sola vez, para proteger contra código 'runaway' que puede desactivar el COP inadvertidamente. Una vez se escribe el registro CONFIG con el COP activado, el COP no se puede desactivar hasta que se haga un Reset en el dispositivo.



El bit COPRS de selección de velocidad del COP en el registro de configuración, selecciona uno de los dos periodos de 'time out'. Poniendo a 1 el bit COPRS se selecciona un periodo de 'time out' del COP de $2^{13} - 2^4$ ciclos CGMXCLK. Con el bit COPRS puesto a 1, un cristal de 32.768-kHz da un periodo de 'timeout' de 250 ms del COP. Poniendo a 0 el bit COPRS, se selecciona un periodo de 'timeout' del COP de $2^{18} - 2^4$ ciclos CGMXCLK.
El contador del COP, es un contador de 6 bits que cuenta libremente, precedido por un contador de 12 bits del prescaler. Si no se borra por software antes de que expire el periodo de 'timeout', los desbordamientos del contador del COP y genera un reset asíncrono. Para prevenir un reset del COP, se escribe algún valor en el registro de control COPCTL del COP antes de que ocurra un desbordamiento. Esto borra el contador del COP y las etapas del 5 al 12 del prescaler.

Módulo LVI (Low Voltaje Inhibit)

En esta parte se explica como configurar y usar el Módulo LVI para proteger el sistema durante una caída de alimentación, haciendo un reset a la MCU. Cuando el voltaje vuelve a su nivel nominal, la MCU continuará trabajando. Se puede seleccionar el voltaje de disparo para trabajar a 3V y 5V. Este módulo reduce el número de componentes y el costo del sistema.

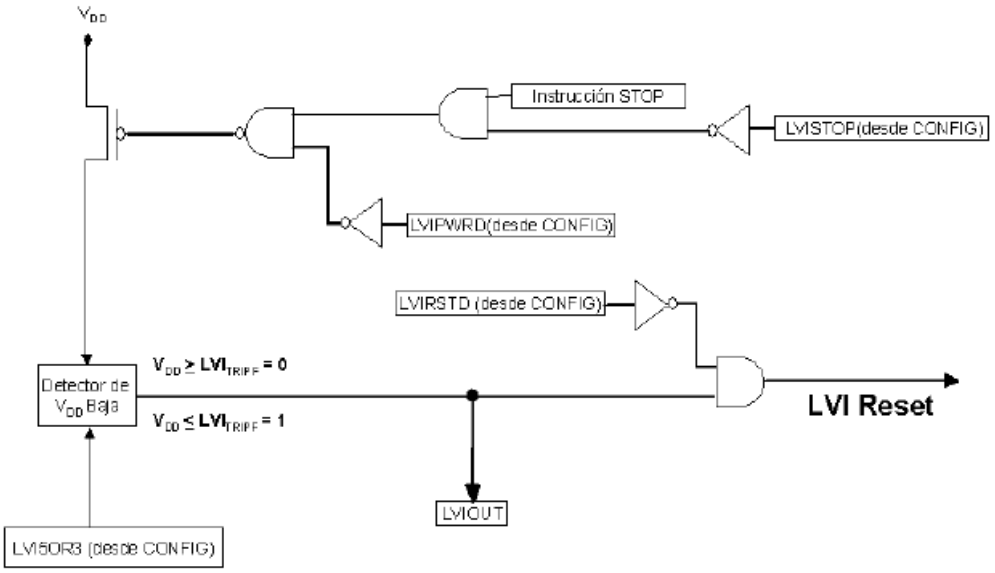
Usos y Características del LVI

- Mejora la fiabilidad del sistema.
- Reduce el número de componentes externos y bajando el costo.
- Hace un reset a la MCU cuando el voltaje baja a partir de un nivel.
- Cuando el voltaje vuelve a su nivel nominal, la MCU continuará trabajando.
- Incluye una selección del voltaje de disparo para sistemas a 3V y 5V.

El funcionamiento de un sistema puede ser afectado por la caída del voltaje de alimentación. Los problemas de alimentación son causas comunes de una condición de ‘brown-out’ (caída de la fuente de alimentación) o de desgaste de una batería que alimenta al sistema. El Módulo LVI protegerá el trabajo de la MCU durante estos eventos.

Diagrama de Bloques del LVI

Ahora, se puede ver cómo se configura este Módulo LVI. El Módulo LVI contiene un circuito de referencia ‘bandgap’ y un comparador que determinan cuando la MCU opera con un voltaje por debajo del punto de disparo especificado. El Módulo LVI se activa fuera del reset y se configura usando varios bits en el registro CONFIG de configuración del sistema. Este registro se inicializa típicamente con el ‘Power-On-Reset’. Una vez se ha escrito el registro no se puede escribir de nuevo hasta que ocurra el siguiente reset. Esta característica de ‘escribir una sola vez’ asegura que una aplicación no puede reconfigurar la MCU inadvertidamente.



- El *bit LVIPWRD* desactiva la alimentación del LVI, también permite al Módulo LVI supervisar el voltaje de alimentación (V_{DD}). Poniendo este bit a 0, se aplica alimentación al circuito analógico LVI. Poniendo a 1 el bit LVIPWRD, se quita la alimentación del Módulo LVI.
- El *bit LVISTOP* activa el modo 'stop' del LVI, determina si el LVI opera en modo 'stop'. Cuando el bit LVIPWRD se pone a 0 y poniendo el bit LVISTOP a 1, permite al Módulo LVI operar durante el modo 'stop'. Un reset borra el bit LVISTOP. La habilidad de desactivar el LVI automáticamente en modo 'stop' permite reducir el consumo del circuito y alarga la vida de la bateria, mientras protege a la CPU durante el funcionamiento normal.
- El *bit LVI5OR3* selecciona el modo de voltaje en que trabaja el Módulo LVI, a 5 V o 3 V. El modo de voltaje en que trabaja el LVI, debe ser igual al voltaje de alimentación (V_{DD}). Poniendo a 1 el bit LVI5OR3, configura el punto de disparo (V_{TRIPF}) para un funcionamiento nominal de 5V. Borrando el bit LVI5OR3, configura el punto de disparo (V_{TRIPF}) para un funcionamiento nominal de 3V. En reset, el LVI tiene un valor predefinido de 3V. Si alimenta un sistema a 5V, después de cada 'power-on-reset'. Los valores del punto de disparo están típicamente en el rango de 4.2 V a 4.5 V en el modo de 5 V a 2.7 V en el modo de 3 V.
- El bit LVIRSTD desactiva la señal de reset del Módulo LVI. Cuando se pone el bit LVIRSTD a 0, el Módulo LVI generará un reset cuando LVIOUT se pone a 1, lo que significa que V_{DD} ha caido por debajo del voltaje de disparo $V_{TRIPF.}$ Si ocurre esta condicion, la MCU permanecera en la condicion de reset hasta que V_{DD} suba por encima del voltaje de disparo V_{TRIPR} V_{TRIPF} es más bajo que $V_{TRIPR,}$ por histéresis, tipicamente de 0.1 V, para evitar entrar y salir de 'reset' cuando el voltaje de alimentación tiene un ligero rizado cercano al voltaje de disparo.

Registro de estado del LVI (LVISR)

El Módulo LVI usa el registro de estado LVISR, para indicar que el voltaje V_{DD} ha caído por debajo del nivel de $V_{TRIPF.}$ El bit LVIOUT del LVI, es un indicador de estado de solo lectura, que el Módulo LVI pone a 1 cuando V_{DD} es menor de $V_{TRIPF.}$ El Módulo LVI pone a 0 este bit cuando V_{DD} sube por encima de $V_{TRIPR.}$ Un reset pone a 0 el bit LVIOUT:

LVISR:

SFEOC	Bit 7	6	5	4	3	2	1	Bit 0
Leer:	LVIOUT	0	0	0	0	0	0	0
Escribir:								
Reset:	0	0	0	0	0	0	0	0

Este indicador de estado es particularmente útil cuando el Módulo LVI se usa en modo de funcionamiento en modo consulta, llamado modo ‘polled’ (no es operativo, solo monitoriza la alimentación y no se puede utilizar para generar un reset). Este modo normalmente se usa en aplicaciones donde se trabaja con niveles de V_{DD} por debajo del nivel de V_{TRIPF}, donde se desea aumentar la vida de la batería, por ejemplo. En este caso, el software puede supervisar consultando (‘polling’) el indicador de estado LVIOUT.
Para configurar el LVI para este modo, hay que activar el Módulo LVI poniendo a 0 el bit LVIPWRD y a 1 el bit LVIRSTD para desactivar resets del LVI.

Módulo SPI (Serial Peripheral Interface)

En esta sección se describen las características y la configuración del Módulo SPI, para trabajar en modo maestro (‘master’) o modo esclavo (‘slave’) y se describe el proceso de transmisión de datos.
El SPI es un módulo de comunicaciones serie, síncrona y full-duplex. Está pensado principalmente para comunicaciones dentro del mismo circuito impreso a velocidades relativamente altas. El SPI es una buena manera de unir via serie la MCU con periféricos externos, como: pequeñas eeproms, convertidores analógico digitales de alta resolución, convertidores digitales analógicos, módulos LCD, controladores de visualizadores a LED de siete segmentos, etc. Comunicando vía serie, la MCU y los periféricos necesitan menos pins y esto revierte a encapsulados mas pequeños y normalmente de menor costo. También el SPI se puede usar para comunicar con otras MCUs.

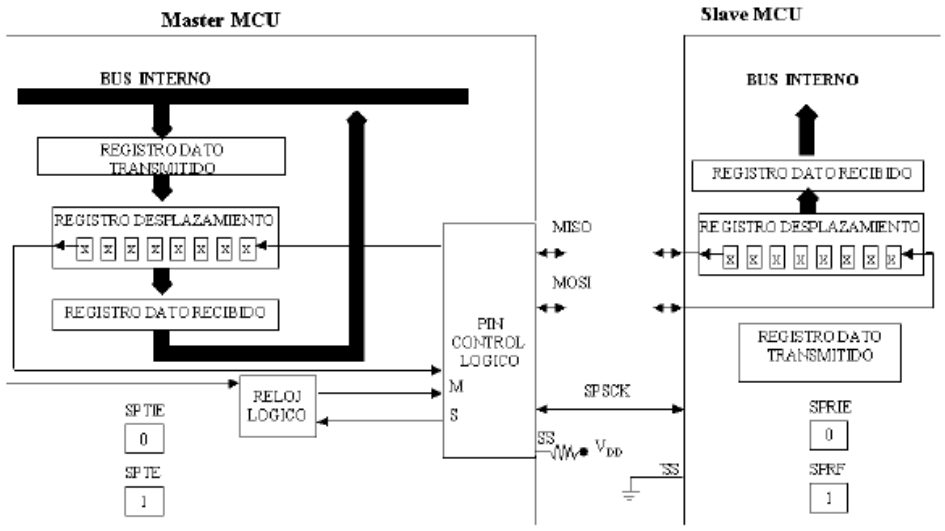
Características:

- Máxima frecuencia a modo ‘master’ = frecuencia de bus / 2
- Máxima frecuencia a modo ‘slave’ = frecuencia de bus
- Reloj Serie con polaridad y fase programable
- Activa dos interrupciones separadas:
  - SPRF (Receptor del SPI completo)
  - SPTE (Transmisor del SPI vacío)
  -

El SPI opera en modo maestro o en modo esclavo. En modo ‘master’, el SPI genera un reloj de comunicación síncrona, a una de las cuatro frecuencias posibles del ‘master’. La frecuencia máxima en modo ‘master’ es de la mitad de la frecuencia del bus. Para la mayoría de MCUs del 68HC08, la frecuencia máxima del bus es de 8 MHz, permitiendo hasta 4 MHz de reloj en modo ‘master’. En modo ‘slave’, el SPI puede operar a velocidades de reloj hasta la frecuencia de bus o hasta 8 MHz, en la mayoría de MCUs del 68HC08.
En el SPI se puede configurar la polaridad y la fase del reloj, permitiendo al SPI comunicar con la mayoría de periféricos serie. Se puede configurar el SPI para generar dos eventos de interrupción separados: transmisor vacío y receptor lleno. Cada interrupción tiene un vector separado que permite transferencias eficaces.

Funcionamiento del SPI

El SPI se ha desarrollado alrededor de un doble registro de desplazamiento ‘buffered’ de 8 bits con ambos extremos del registro de desplazamiento, que van a los pines de la MCU. Un extremo del registro de desplazamiento se conecta al pin MISO (Master-In-Slave-On). Este pin actúa como una entrada para el Módulo ‘master’ del SPI y como una salida para el Módulo ‘slave’ del SPI. El otro extremo del registro de desplazamiento se conecta al pin MOSI (Master-On-Slave-In).



La CPU empieza una transferencia serie SPI, escribiendo un byte de datos en el registro de datos transmitidos. Se transferiran automáticamente via serie todos los 8 bits de datos a traves del pin MOSI del 'master', sincronizados con el reloj de salida del 'master' (SPSCK). Cada vez que se desplaza un bit a traves del pin MOSI del 'master', se desplaza un bit a traves del pin MISO del 'master', permitiendo una comunicación 'full-duplex'.

Registro SPCR de control del SPI

SPCR	Bit 7	6	5	4	3	2	1	Bit 0
Leer:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
Escribir:								
Reset:	0	0	1	0	1	0	0	0

- El *bit SPRIE* de interrupción del receptor, es un bit de lectura/escritura que activa las peticiones de interrupción. Cuando el bit SPRIE se pone a 1, se genera una interrupción cuando el bit SPRF receptor del SPI se pone a 1. el bit SPRF se pone a 1 cuando se transfiere un byte de datos al registro de datos del SPI.

- El *bit DMAS* de selección de DMA, es un bit de solo lectura que no tiene efecto en las MCU 68HC80 que no tienen un controlador de DMA.

- El *bit SPMSTR* 'master' del SPI, es un bit de lectura/escritura que selecciona el modo de trabajo del SPI. Poniendo el bit a 1 para seleccionar el modo 'master' y a 0 para seleccionar el modo 'slave'.

- El *bit CPOL* de polaridad del reloj, es un bit de lectura/escritura que determina el estado lógico del bit SPSCK entre las transmisiones. Para transmitir datos entre dos módulos SPI, los módulos SPI deben tener valores idénticos del bit CPOL.
- El *bit CPHA* de la fase del reloj, es un bit de lectura/escritura que controla la relación de tiempo entre el reloj serie y los datos del SPI. Para transmitir datos entre dos módulos

SPI, deben tener valores idénticos de CPHA. La fase y la polaridad se podrán ver mas adelante.

- El *bit SPWOM* modo OR alambrado, es un bit de lectura/escritura que desactiva los 'pullup' en los pins SPSCK, MOSI y MISO para que estos pins se pongan en salida 'open-drain'.

- El *bit SPE* activa el SPI, es un bit de lectura/escritura que activa el módulo SPI. Se debe desactivar el SPI antes de que cambie la fase o la polaridad del reloj, escribiendo el bit CPOL o el bit CPHA.

- El *bit SPTIE* activa la interrupción de transmisión, es un bit de lectura/escritura que activa las peticiones de interrupción de la CPU que se vayan a generar cuando se transfiere un byte desde el registro transmisor de datos al registro de desplazamiento.

Registro de estado y de control del SPI (SPSCR)

El registro SPSCR de estado y de control del SPI, contiene el indicador de estado y los bits de control adicionales.

SPSCR	Bit 7	6	5	4	3	2	1	Bit 0
Leer:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
Escribir:								
Reset:	0	0	0	0	1	0	0	0

- El *bit SPRF* de receptor lleno del SPI, es un bit de solo lectura que se pone a 1 cada vez que transfiere un byte del registro de desplazamiento al registro de datos del receptor. Si el bit SPRIE en el registro SPCR se pone a 1, poniendo a 1 el bit SPRF genera una petición de interrupción a la CPU. Durante una interrupción de la CPU, la CPU pone a 0 el bit SPRF leyendo el estado del SPI y el registro de control, con el bit SPRF puesto a 1 y después leyendo el registro de datos del SPI.

- El *bit ERRIE* activa la interrupción de error, es un bit de lectura/escritura que le permite al SPI generar interrupciones en condiciones de error para tomar acciones correctivas. Las dos condiciones de error son: desbordamiento y modo fallo.

- El *bit OVRF* de desbordamiento, es un bit de lectura/escritura que se pone a 1 cuando el software no lee el byte en el registro de datos del receptor antes de que el siguiente byte completo entre en el registro de desplazamiento. En una condición de desbordamiento, el último byte que se ha desplazado se pierde. Se puede poner a 0 el bit OVRF leyendo el registro de estado y de control con el bit OVRF puesto a 1, leyendo el registro de datos del receptor.
- El *bit MODF* de modo fallo, es un indicador de solo lectura que se pone a 1 cuando ocurre un error de fallo durante la transmisión y el bit MOFDEN que activa el modo de fallo se pone a 1. Las condiciones bajo las que el indicador MODF se pone a 1, son diferentes para el 'master' y el 'slave'. En un SPI 'master', el bit MODF se pone a 1 siempre que el pin SS pasa a valor bajo. En un SPI 'slave', el bit MODF se pone a 1 cuando el pin SS pasa a valor alto, durante la transmisión.

- El *bit SPTE* de transmisor vacío, es un bit de lectura/escritura que se pone a 1 cada vez que el registro de datos de transmisión transfiere un byte al registro de desplazamiento. Poniendo a 1 el bit SPTE, genera una interrupción si es activado por el bit SPTIE. No se debe escribir en el registro de datos del SPI a menos que el bit SPTE se ponga a 1.

- Los *bits SPR1 y SPR0* de selección de ‘baud rate’, son bits de lectura/escritura que selecciona uno de los cuatro divisores de velocidad de transmisión. Puesto que el reloj solo se maneja en modo ‘master’, estos bits no tienen efecto en modo ‘slave’.

Módulo ADC (Analog to Digital Converter)

En esta parte se describen las características y la configuración del Módulo ADC. Así como las técnicas para obtener la máxima precisión en las conversiones del ADC.

Características del ADC

Todos los Módulos ADC soportan dos modos de conversión, en modo de conversión continua y en modo de conversión única. En modo de conversión única, se completa una conversión entre escribir el registro de estado y el registro de control. En modo de conversión continua, la entrada analógica del ADC convierte continuamente y lo escribe en el registro de datos del ADC. En este modo, los datos de la conversión anterior se borran sin tener en cuenta si estos datos han sido leídos o no.

Registro ADSCR de estado y de control

ADSCR

$0003C	Bit 7	6	5	4	3	2	1	0
Leer: / Escribir:	COCO/ IDMAS	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
Reset:	0	0	0	1	1	1	1	1

- El indicador *COCO*, es un bit de solo lectura e indica cuando una conversión se ha completado. En modo de una sola conversión, el ADC pone el indicador COCO a 1, después de que cada conversión haya sido completada. En modo de conversión continua, el ADC pone el indicador COCO a 1, después de que la primera conversión haya sido completada.
- En el 68HC08 que tiene el módulo de acceso directo a memoria (DMA), se puede usar el bit 7 para controlar el funcionamiento del DMA. En este tipo de configuración, el bit 7 ADSCR se refiere a IDMAS. Solo se debe escribir en esta posición si el dispositivo tiene DMA. Escribiendo en esta posición, en las MCU que no contienen DMA, se enmascararán las interrupciones del ADC y causará resultados no deseados.
- Se pone el indicador *AIEN* a 1 para configurar el ADC para generar una señal de interrupción cuando la conversión se haya completado. La señal que genera la interrupción se pone a 0, cuando el registro de datos se lee o se escribe el registro de estado y de control.
- Se puede seleccionar el modo de conversión con el indicador *ADCO* de conversión continua. Poniendo este bit a 1 se selecciona el modo de conversión continua. Cuando este bit se pone a 0, el ADC se configura en modo de conversión simple.
- Los 5 bits restantes del ADSCR, *ADCH4-ADCH3-ADCH2-ADCG1-ADCH0* del ADSCR, contienen los bits de selección de los canales del ADC.

Registro ADR de datos del ADC

El registro de datos ADR del ADC es un registro de solo lectura que el ADC usa para guardar los resultados de la conversión. El ADC actualiza el registro ADR después de que haya sido completada cada conversión.

ADR	Bit 7	6	5	4	3	2	1	Bit 0
Leer:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
Escribir:								
Reset:	0	0	0	0	0	0	0	0

Registro ADCLK del reloj del ADC

Usando los bits del prescaler del reloj del ADC, ADIV2-ADIV1-ADIV0, se puede seleccionar uno de los cinco valores del divisor: 1, 2, 4, 8 o 16. El ADC genera la frecuencia de reloj dividiendo la fuente de reloj por el valor del divisor seleccionado.

ADCLK	Bit 7	6	5	4	3	2	1	Bit 0
Leer:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
Escribir								
Reset:	0	0	0	0	0	0	0	0

Se puede seleccionar la fuente del reloj de entrada del ADC, usando el bit ADICLK de selección de reloj del ADC. Si este bit se pone a 0, el ADC usará el reloj externo CGMXCLK como reloj de entrada. Si se pone a 1, el ADC usará el reloj del PLL interno del 68HC08 como reloj de entrada.
Se debe seleccionar la fuente de reloj de entrada basandose en la frecuencia del reloj. El módulo del ADC se ha diseñado para operar mejor con una frecuencia de reloj de entrada de 1 MHz. Si la frecuencia de reloj externa es mayor o igual a 1 MHz, se debe usar el reloj externo como fuente de entrada. Por otra parte, hay que usar el reloj interno del bus como fuente de entrada.

Cálculo del tiempo de una conversión

Una vez se ha seleccionado la fuente de reloj de entrada, se puede calcular la cantidad de tiempo que se usa para completar una sola conversión. Primero se determina el número de ciclos de reloj que se usa para completar la conversión y entonces dividir este valor por la frecuencia de reloj de entrada.

$$\text{Tiempo de una Conversión} = \frac{\text{16 o 17 (ciclos del ADC)}}{\text{1MHz (Frecuencia del ADC)}} = 16 \text{ o } 17\,\mu\text{s}$$

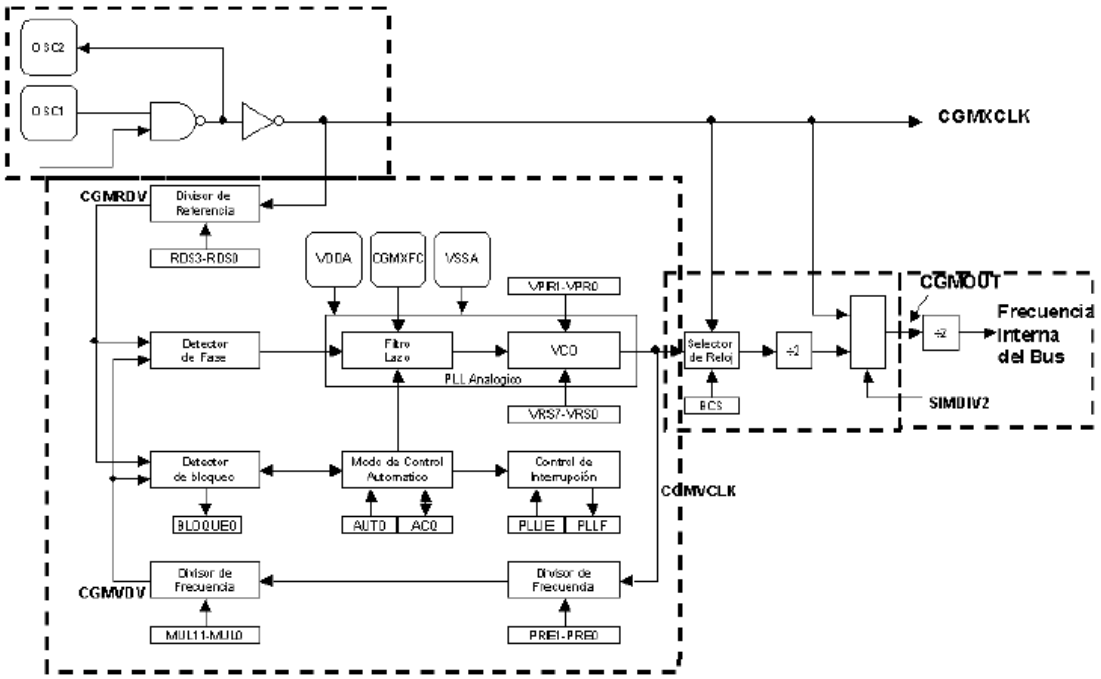
El proceso de conversión empieza después de que se ha escrito el registro ADSCR. Para completar una conversión típica usa 16 ciclos de reloj del ADC. Cuando hay un retraso de sincronización del reloj, entre el reloj de la CPU y el reloj del ADC, para completar la conversión usará 17 ciclos de reloj. Es posible un retraso de sincronización del reloj si el reloj del A/D es diferente que el reloj de la CPU. Con la frecuencia de reloj de entrada puesta a 1 MHz, una conversión típica usará aproximadamente de 16 a 17 µs.

Módulo Generador de Reloj

En este Módulo Generador de Reloj (CGMC) del 68HC08, se describen las características y su configuración. Se selecciona una fuente de reloj base apropiada, se calculan los diferentes valores requeridos para programar y se configura el PLL, para generar una frecuencia de bus específica.
El CGMC incluye un PLL integrado que es capaz de generar frecuencias en múltiplos enteros del cristal de referencia. Por ejemplo, usando el PLL se puede proporcionar hasta una frecuencia de bus de 8.2 MHz usando un cristal de 32 KHz, de muy bajo costo. El uso del PLL reduce la generación de ruido, en comparación con los cristales de frecuencias más altas y no son necesarios los osciladores.
El módulo CGMC tiene dos prescalers programables, que aumentan por dos la frecuencia. También incluye un hardware con un oscilador programable por tensión (VCO) para un funcionamiento con bajas fluctuaciones (jitter). El Módulo CGMC puede cambiar automáticamente del modo ‘adquisición’ al modo ‘rastreo’ (tracking), dependiendo de cuanto esté desplazada la frecuencia de salida con respecto a la frecuencia de referencia.
El PLL detecta automáticamente cuando la frecuencia de salida está bastante cerca de la frecuencia deseada y se bloquea (permanecerá en esa frecuencia). La CPU se puede configurar para generar una interrupción cuando la frecuencia deseada se bloquea o sale de la condición de bloqueo. En el registro CONFIG, se puede poner a 1 el bit OSCSTOPENB para permitir que el oscilador opere durante el modo STOP.

Diagrama de Bloques del CGMC

El CGMC consiste en tres submódulos: el circuito del oscilador a cristal, el PLL y el circuito de selección de reloj base.

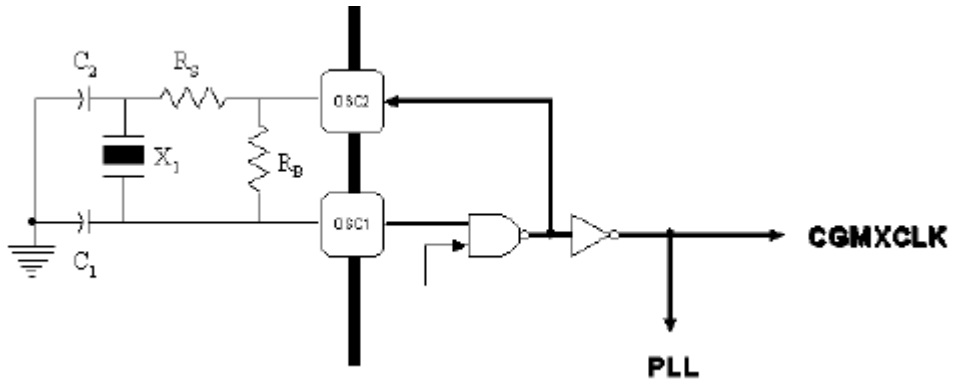


El circuito del oscilador a cristal consiste en un amplificador inversor y un cristal externo. El pin OSC1 es la entrada del amplificador inversor y el pin OSC2 es la salida. El circuito del oscilador a cristal genera la frecuencia de reloj del cristal constante, CGMXCLK corre a una velocidad igual a la frecuencia del cristal. CGMXCLK pasa por un 'buffer' para producir la referencia de reloj del PLL.
El PLL es un generador de frecuencia totalmente funcional, que ha estado diseñado para su uso con cristales o con resonadores cerámicos. El PLL genera la frecuencia de reloj programable CGMVCLK con el VCO. La frecuencia del VCO será un múltiplo entero de la frecuencia de referencia. El circuito de selección del reloj base CGMOUT se controla por software. Se puede poner el reloj base a CGMXCLK dividido por dos o CGMVCLK dividido por dos.
El Módulo SIM (Módulo Integración del Sistema) posteriormente deriva los relojes del sistema desde CGMOUT o CGMXCLK, con una división adicional a través de dos circuitos, para producir la frecuencia interna del bus.

Características del Oscilador a Cristal

El CGMXCLK puede ser usado por módulos que requieren trabajar con unos tiempos muy precisos. El ciclo de servicio CGMXCLK no se garantiza que sea del 50% y depende de factores externos, incluyendo el cristal y otros componentes externos. También se puede usar un reloj generado externamente entrando por el pin OSC1 del circuito oscilador a cristal. Para usar esta configuración, hay que conectar el reloj externo al pin OSC1 y dejar el pin OSC2 al aire.
El circuito oscilador a cristal puede tomar dos fuentes de frecuencia diferentes, la de un cristal o la de un oscilador externo. Si se elige usar un cristal, la frecuencia del cristal debe estar entre 30 KHz y 100 KHz, típicamente de 32.768 KHz. El oscilador no se ha diseñado para frecuencias de cristal fuera de este rango.
Si se elige usar un oscilador externo como fuente de reloj, se puede elegir tener el PLL activado o se puede desactivarlo. Con el PLL activado, la frecuencia del oscilador externo debe estar entre 30 KHz y 1.5 MHz. Con el PLL desactivado, la frecuencia del oscilador externo puede ir de cero a 32 MHz.

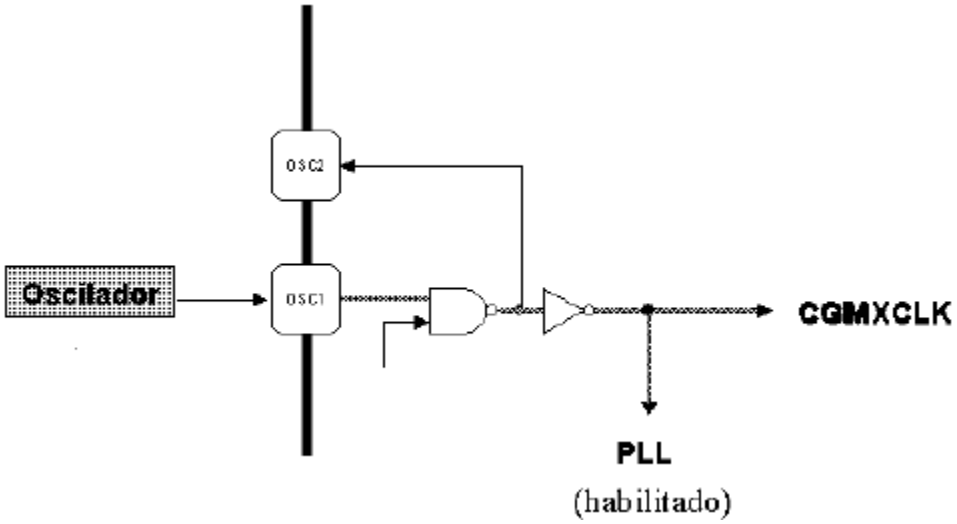
Circuito Oscilador a Cristal



Al usar un cristal como fuente de reloj, la frecuencia se puede seleccionar como un reloj base o como un reloj de referencia para el PLL. En esta configuración, el PLL se puede activar o se puede desactivar. La frecuencia del cristal debe estar entre 30 KHz y 100 KHz, para garantizar un buen funcionamiento.

Oscilador Externo

Con la configuración de oscilador externo, se puede trabajar con el PLL activado o desactivado dependiendo de la frecuencia del oscilador.



- Con el PLL activado, se puede usar un oscilador externo con una frecuencia entre 30 KHz y 1.5 MHz.
- Con el PLL desactivado, se puede usar un oscilador externo con una frecuencia entre DC y 32.8 MHz.

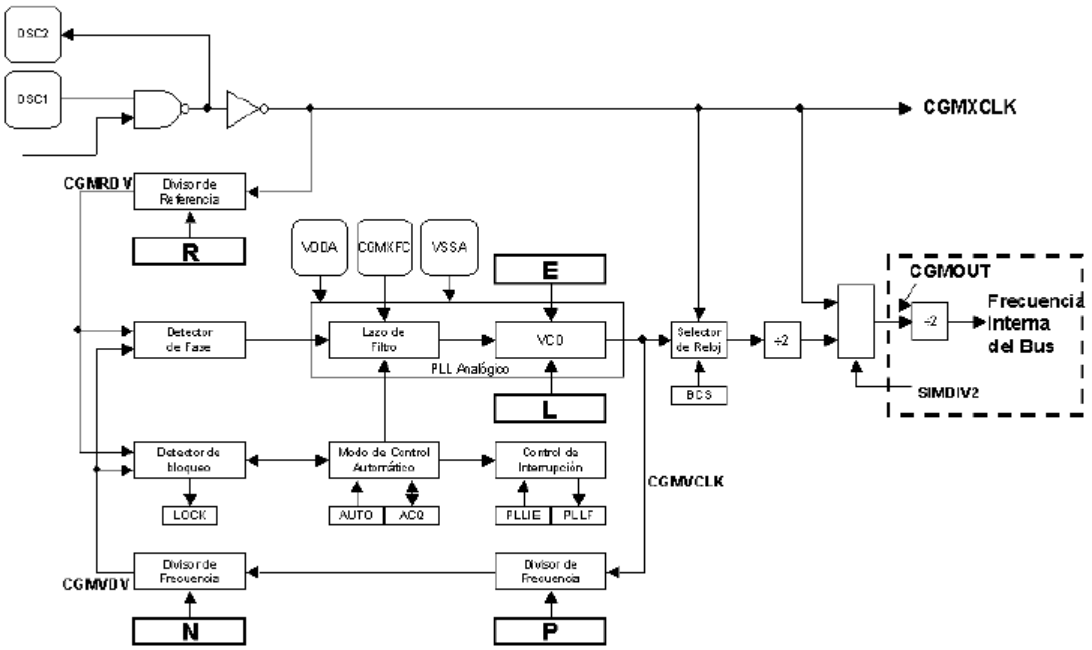
Características del PLL

Como se vio antes, el PLL genera la frecuencia del VCO programable. El PLL puede filtrar la frecuencia del VCO, usando dos modos: modo adquisición y modo rastreo. La selección del modo, depende de la exactitud de la frecuencia de salida.
En *modo adquisición*, el filtro puede hacer correcciones de frecuencia grandes a la frecuencia del VCO. Este modo se usa al iniciar el PLL o cuando el PLL ha sufrido un ruido severo y la frecuencia del VCO está desplazada de la frecuencia deseada. Para seleccionar este modo, hay que seleccionar el bit ACQ en el registro de control del ancho de banda del PLL.
En *modo rastreo*, el filtro hace pequeñas correcciones a la frecuencia del VCO. Las fluctuaciones del PLL son muy bajas, pero la respuesta al ruido también es más lenta. El PLL entra en modo rastreo cuando la frecuencia del VCO es casi correcta, como cuando el PLL se selecciona como fuente de reloj base.
El PLL puede cambiar entre el modo adquisición y modo rastreo, automática o manualmente. Se recomienda el modo automático para la mayoría de las aplicaciones. Hay muchas ventajas al incluir un PLL en la MCU. Se puede lograr una frecuencia de bus alta con un cristal económico, ya que los cristales de baja frecuencia son más baratos. Los cristales de baja frecuencia también consumen menos y proporcionan más alta inmunidad al ruido.

Módulo Generador de Reloj

El CGMC usa varios factores para programar el PLL. **R** es el valor para el divisor de referencia. **E** es el valor del VCO, con un rango multiplicador potencia de dos. **L** es el multiplicador con

rango lineal del VCO. **P** es el divisor de frecuencia para el prescaler y **N** es el valor del modulo divisor de frecuencia del VCO.

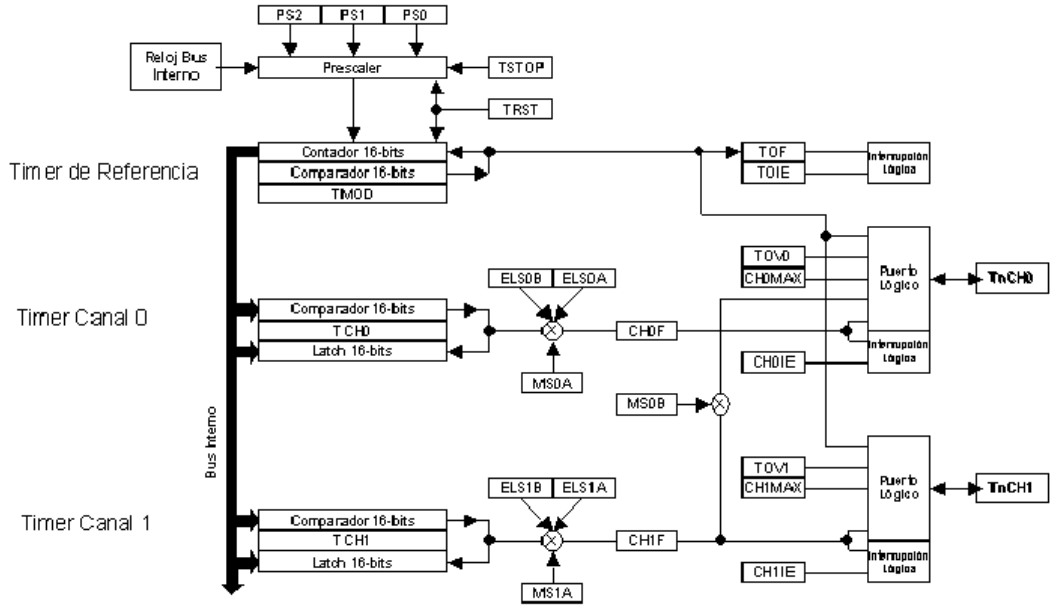


Módulo TIM (Timer Interface Module)

Esta parte cubre la configuración y el uso en detalle del TIM. Prepara y realiza una función de comparación de salida (‘output compare’), una función de captura de entrada (‘input capture’) y algunas funciones de modulación de ancho de pulso (PWM), mostrando las diferencias entre el PWM ‘unbuffered’ (con perdida de tiempo en la recarga) y PWM ‘buffered’ (sin perdida de tiempo por el valor cargado en un segundo registro).

Diagrama de Bloques del TIM

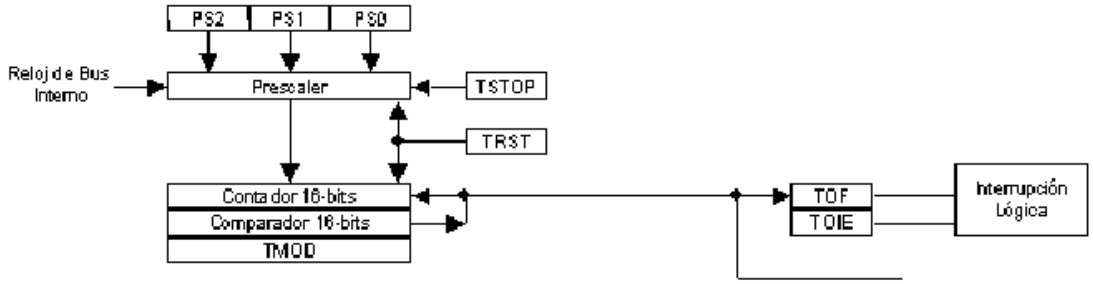
El TIM del MC68HC908JK1/JK3 tiene los canales del ‘Timer’ seleccionables por el usuario, en lugar de funciones especificas del ‘input capture’ (captura de entrada) y ‘output compare’ (comparación de salida). Cada canal es programable por el usuario para realizar una función de comparación de salida, una función de captura de entrada o una función de modulación de ancho de pulso. Estas funciones proporcionan flexibilidad para configurar el ‘Timer’ exactamente como se necesita para cada aplicación específica. Tambien se pueden reconfigurar los canales en una aplicación para realizar otras tareas diferentes.
Cada 68HC08 tiene implementado con un módulo TIM con 2, 4, 6 u 8 canales de ‘Timer’ programables. Este diagrama de bloques solo tiene dos canales TIM.



Cada canal del ‘Timer’ comparte una sola referencia de tiempo, que viene de un módulo contador de 16-bits, un prescaler y un comparador. El prescaler es programable y proporciona, al contador común de referencia de 16 bits, un reloj que se deriva de la frecuencia del bus y que se divide por una de las siete posibilidades.
El módulo comparador permite al timer de 16-bits, contar solo hasta el valor cargado en el registro del módulo comparador, TMOD, antes de un reset y reiniciando otra secuencia del conteo.
Se pueden obtener interrupciones por desbordamiento del contador y por cualquier actividad del canal del timer de entrada o de salida. Cada fuente de interrupción tiene su propio vector de interrupción, para permitir atender una interrupción sin perder tiempo analizando la fuente de la interrupción.

Tiempo de Referencia

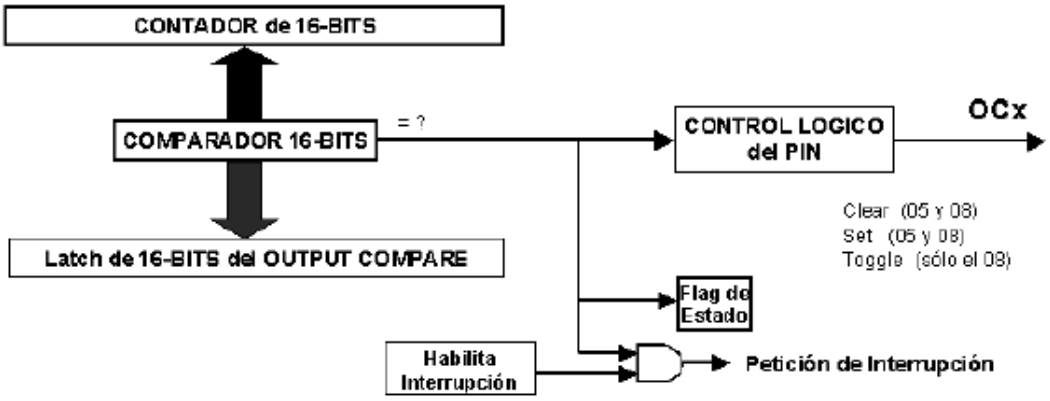
La referencia de tiempo se genera por un contador de 16-bits, un prescaler y un módulo comparador. Para el trabajo normal, el contador de 16-bits corre continuamente como un contador libre. El contador se puede detener o se puede restablecer bajo el control del programa. Esto es particularmente útil cuando se quiere sincronizar la referencia del TIM con otro 'timer' o reloj.
El prescaler proporciona un valor de reloj derivado del bus interno del 68HC08, al contador de 16-bits y de un divisor programable. Usando PS0, PS1 y PS2, se puede programar el divisor por uno de los siete valores: divido por 1, por 2 o por cualquier otra división binaria hasta por 64. También se puede seleccionar un reloj externo.
En su modo de mas alta resolución (divido por 1), el TIM tiene una resolución de 125ns cuando está operando a la frecuencia de bus interno normal de 8 MHz (1 dividido por 8 MHz). La referencia de reloj de 16 bits a una proporción más lenta, el prescaler puede dividir a menor frecuencia del bus interno, tanto como divido por 64.



La referencia de tiempo de 16-bits incluye un registro TMOD, que se compara con el contador de 16-bits. Cuando se igualan, el indicador TOF de desbordamiento del 'timer' se pone a 1 y el contador de 16-bits se restablece y empieza otra secuencia de conteo. Por defecto, el registro del módulo se pone a *FFFhex* que permite a la referencia contar a través de los 16-bits.
Se pueden cronometrar eventos mas largos de 65,536 ciclos (16-bits), usando la función de desbordamiento del 'timer'. Cuando el indicador TOIE de interrupción de desbordamiento del 'timer' se pone a 1, se generará un evento de interrupción cada vez que el indicador TOF se pone a 1. Para cronometrar un evento de larga duración, se puede escribir código para contar el número de desbordamientos que ocurren. El prescaler mantiene la capacidad de optimizar la resolución versus el proceso de desbordamiento del 'timer'.

Función 'Output Compare'

La función de comparación de salida usa la referencia de tiempo de 16-bits, un comparador, un 'latch' de 16-bits de comparación de salida (output compare), un pin de salida, un pin de control lógico y una lógica de interrupción. La función de comparación de salida se puede ver con un ejemplo, de simple retardo.



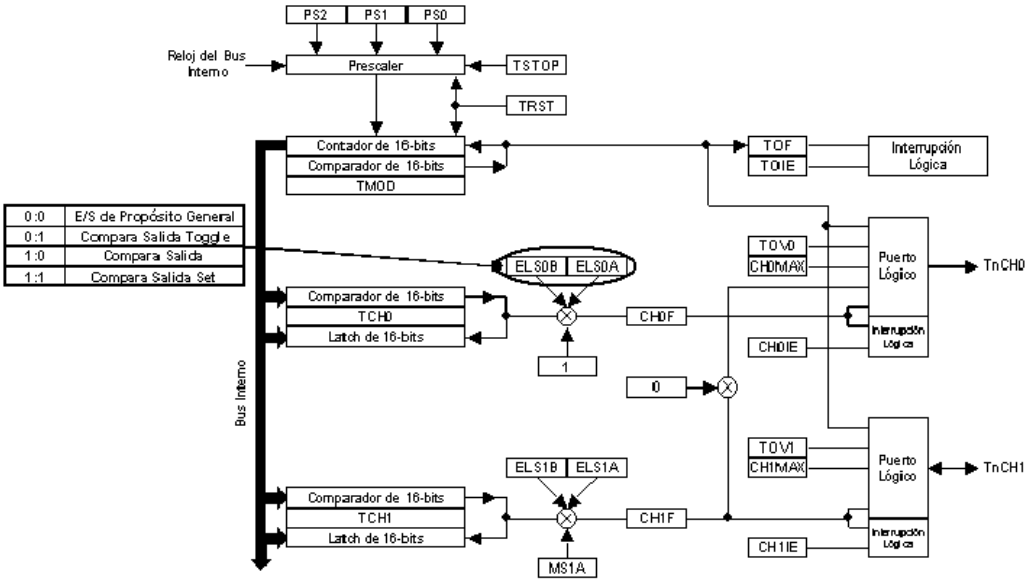
Se empieza el ejemplo activando las interrupciones. Después, se selecciona un valor de 'offset' basado en el retardo deseado y se añade a este el valor del contador que corre libremente de 16-bits. Se escribe la suma del valor del contador y el 'offset' en el pin de comparación de salida. Cuando el contador que corre libremente se iguala al 'latch' de comparación de salida, se pone a 1 un indicador de estado, indicando que el retardo cronometrado ha ocurrido.
Opcionalmente, se puede poner un pin de salida a 1, a 0 o conmutando ('toggle') cuando el contador se iguala al valor de comparación del 'latch'. Se puede generar una interrupción opcional, permitiendo al usuario acercarse a tiempos muy precisos y crear eventos externos usando los pins de entrada/salida de comparación.

Aplicaciones 'Output Compare'

Se puede usar la función de comparación de salida, para una variedad de controles, incluyendo funciones comunes de cronometraje. En el ejemplo anterior, se usa la función de comparación de salida para realizar un simple retardo. Se puede extender el ejemplo, para realizar una interrupción periódica activando la interrupción de comparación de salida. Cuando la rutina de servicio recibe una interrupción de comparación de salida, calcula un nuevo valor de comparación de salida antes de continuar el proceso normal.
Usando el pin de salida puesto a 1, a 0 y opcionalmente conmutando ('toggle'), se puede implementar un simple pulso con ancho variable. Usando las mismas técnicas, se puede generar una frecuencia de salida variable o una señal de modulación de ancho pulso (PWM).

Aplicación de la Función 'Output Compare'

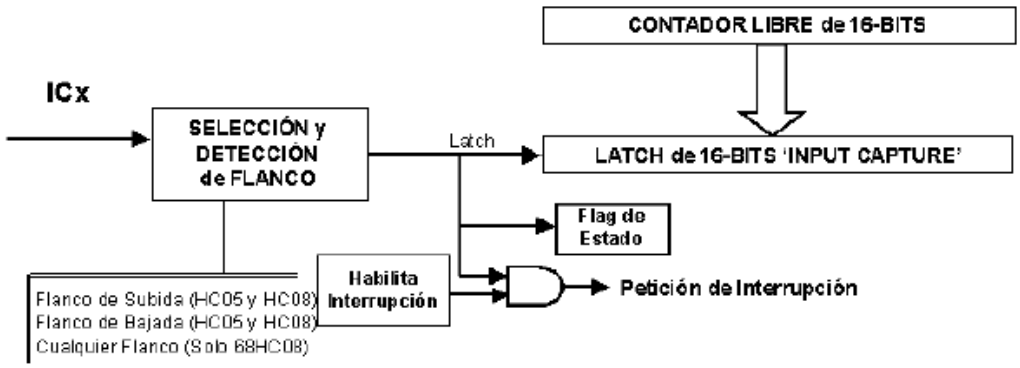
Se puede ver un ejemplo más detallado de una función de comparación de salida (output compare) usando el Canal 0. Se empieza poniendo MS0B a 0 para aislar los dos canales del timer y poniendo MS0A a 1 para seleccionar la función de salida. Entonces se utiliza ELS0A y ELS0B para programar la operación del pin.



La opción ‘toggle’ se puede usar para simplificar la generación de pulsos. La comparación de salida (output compare) se usa para generar señales temporizadas. El valor guardado en TCH0 se está constantemente comparando con el contador de 16-bits. Cuando el valor en el contador de 16-bits se iguala a TCH0, la transición de la señal de salida programada (ELS0B:ELS0A) se fuerza en el pin TxCH0. Esto también puede generar una interrupción del canal del timer, si estaba activada.
Tópicamente, se usará una rutina de servicio de interrupción, para fijar la próxima señal de salida del timer. Simplemente se lee el valor en TCH0, se añade un tiempo de ‘offset’ y se guarda el resultado en TCH0. Una vez más, en cuanto el valor del contador se ponga igual a TCH0, ocurre la actividad programada. Se da por supuesto que el contador corre libremente, que cuenta hasta *FFFFhex* y vuelve a cero. Si no, se tiene que tener en cuenta esto, al calcular el nuevo valor para escribir a TCH0.

Aplicación de la Función ‘Input Capture’

La función de captura de entrada, se hace a través de un pin de entrada con selección de flanco, detector lógico y lógica de interrupción, usa el contador de 16-bits que corre libremente y el ‘latch’ del ‘input capture’ de 16-bits. Esta función permite cronometrar eventos externos para ser referenciados al contador que corre libremente de 16-bits.



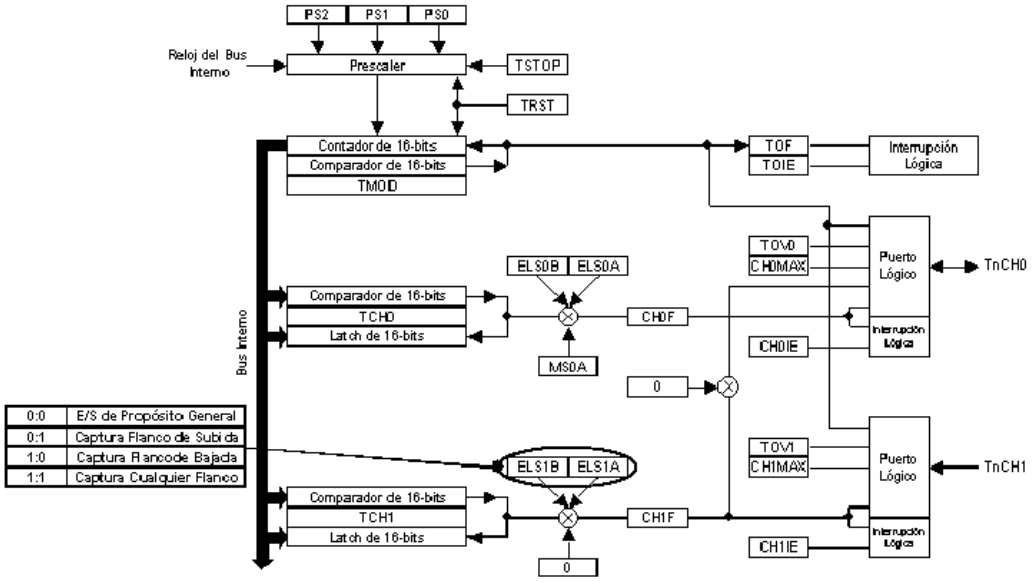
Se puede configurar el pin de la entrada para buscar un flanco de subida, un flanco de bajada o cualquier tipo de flanco. En este ejemplo se selecciona el flanco de subida. También se activan las interrupciones. Una vez que se ha detectado el flanco seleccionado, el latch del 'input capture' se carga con el valor del contador que corre libremente y graba un tiempo de cuando ocurrió el evento. Se pone a 1 el indicador de estado y se puede generar una interrupción opcionalmente.

Aplicaciones 'Input Capture'

Se puede usar la función de captura de entrada ('input capture') para una variedad de funciones de control. En el ejemplo anterior, se usa la función de captura de entrada ('input capture') para realizar una referencia de tiempo absoluto de un evento externo. También se puede medir un periodo de entrada, activando la interrupción de captura de entrada. Cuando la rutina de servicio recibe una interrupción de captura de entrada, guarda temporalmente una copia del 'latch' de captura de entrada y después vuelve al proceso normal hasta que el flanco seleccionado es detectado de nuevo. La rutina de servicio substrae los dos valores de la captura de entrada para determinar el periodo. Para periodos largos, la rutina de servicio se ajusta para el número de interrupciones de desbordamiento del timer que ocurren entre el primer y segundo flanco.
Usando el 'input capture', se puede medir el ancho de un pulso de entrada. Esta función es similar a medir un periodo, excepto que la segunda captura se configura para detectar el flanco de bajada en lugar del flanco de subida inicial. Para anchos de pulso muy cortos, se pueden usar los dos canales del timer para mirar la misma señal con un canal para detectar el flanco de subida y el otro canal para detectar el flanco de bajada. Esto permite hacer medidas por debajo de 125 ns.
Si no se necesita la función de captura de entrada, se puede usar el pin de captura de entrada, como una línea de interrupción adicional.

Aplicación de la Función 'Input Capture'

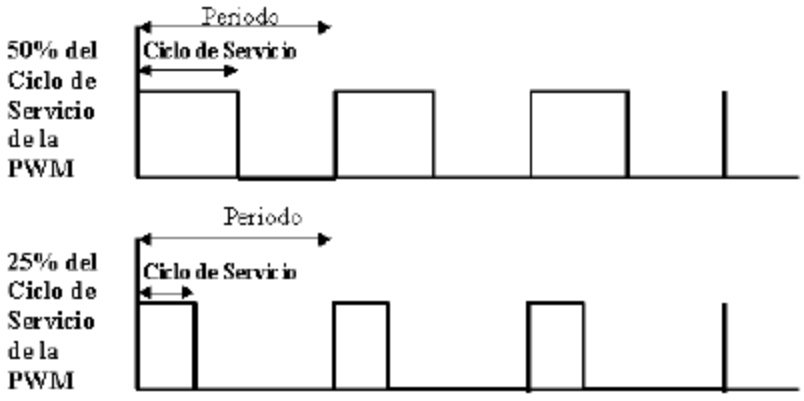
Se usa el Canal 1 para realizar una captura de entrada ('input capture'). Una vez más, se pone MS0B a cero, para aislar los Canales 0 y 1. También se pone MS1A a cero para seleccionar la función de entrada. ELS1A y ELS1B se usan para programar la operación del pin, se puede elegir cualquier forma de capturar un flanco, simplificando la medida del ancho de pulso, reduciendo el software.



Se usa el 'input capture' para medir señales cronometradas. El registro del canal del timer TCH1 se usa para poner el valor del 'latch' en el contador de 16-bits. Cuando ocurre la transición programada (ELS1B y ELS1A) en el pin (TxCH1), el valor del contador de 16-bits se guarda en el registro del canal del timer TCH1. Se puede generar opcionalmente una interrupción en la transición. Típicamente, se usa una rutina de servicio de interrupción para medir y calcular el tiempo entre transiciones de una señal, donde se determina el pulso o el ancho del periodo. Sabiendo el valor de dos transiciones, se puede calcular el pulso o el ancho del periodo.

Función PWM

La modulación del ancho de pulso, se usa para generar una forma de onda con un periodo fijo y el ciclo de servicio variable. El ciclo de servicio, es el tiempo relativo utilizado en los estados altos y bajos del periodo de la señal. Los moduladores de ancho de pulso pueden tener diferentes frecuencias y resoluciones. La frecuencia está definida por el periodo y la resolución es definida por el número de pasos discretos del ciclo de servicio que se pueden poner a 1.

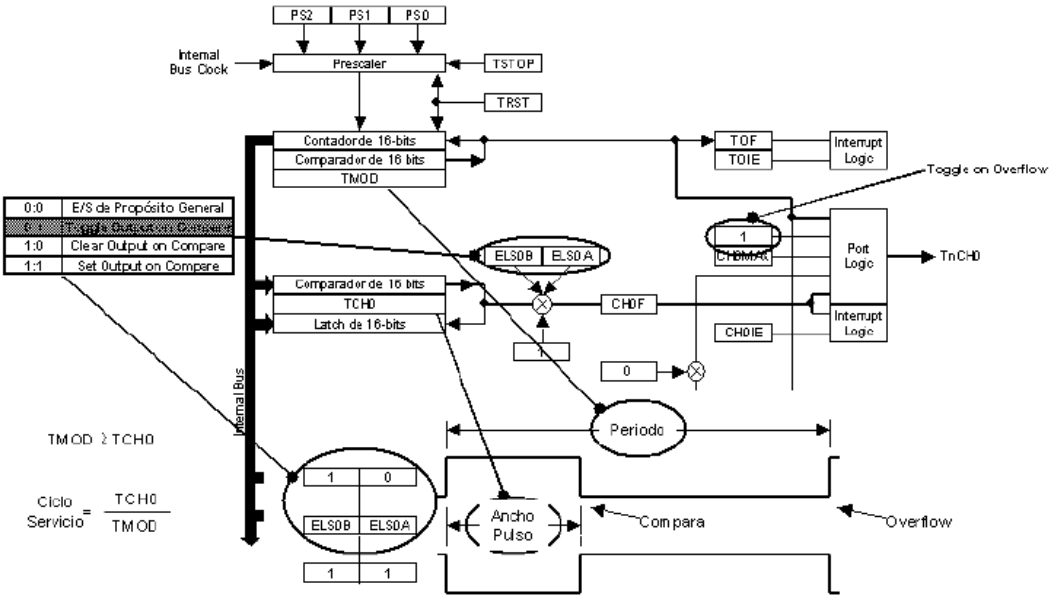


Un PWM de 8-bits permite especificar el ciclo de servicio en 256 pasos, es decir 2 elevado a 8. Por ejemplo, un ciclo de servicio de no-cero tan pequeño como 0.39% o lo que es lo mismo que 1 dividido por 256, podría especificarse con un PWM de 8-bits.

Un uso común del modulador del ancho de pulso, es la conversión digital a analógico usando algún filtro externo. El voltaje analógico generado es proporcional al ciclo de servicio. Teóricamente, un 50% del ciclo de servicio generará aproximadamente la mitad del voltaje analógico máximo y un 25% del ciclo de servicio generará un cuarto del voltaje analógico máximo.
El modulador de ancho de pulso también se usa para controlar normalmente un motor y para controlar la carga de una batería.

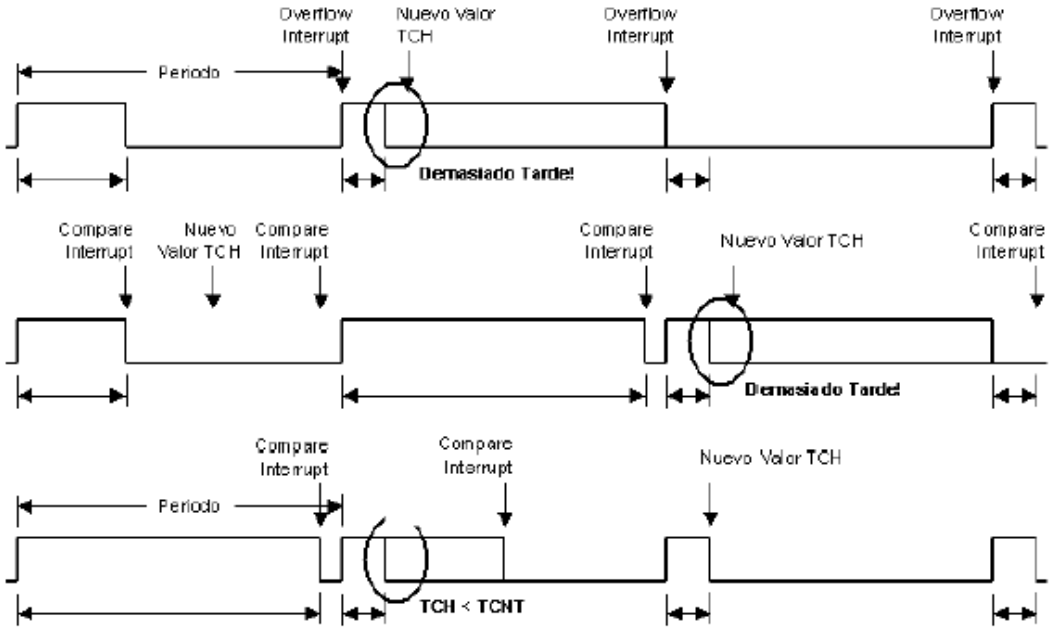
Función PWM 'Unbuffered'

Primero, se puede ver el procedimiento de puesta a punto e implementación de un PWM 'unbuffered' y después de un PWM 'buffered'. También se verán las diferencias entre los dos tipos.



El PWM 'unbuffered' opera como la función 'output compare'. Se muestra usando el Canal 0 del Timer. Así como la función 'output compare', si se pone MS0B a 0 se aíslan los dos canales de los otros y si se pone MS0A a 1 se selecciona la función de salida. Por semejanza, se usa ELS0A y ELS0B para programar el funcionamiento del pin.
Hay que recordar que una señal PWM consiste en un Periodo y en un Ancho de Pulso. Para mayor flexibilidad, estos dos parámetros pueden ser programables.
Para generar la señal como la que se muestra en la figura, se configurará ELS0B:ELS0A como 1:0 o lo que es lo mismo "Borrar la Salida en Comparación". Después se usa el Registro del Canal del Timer (TCH0) para determinar el ancho del pulso.
El Registro del Módulo Contador determina el periodo, obligando un desbordamiento del contador y restableciendo el Registro Contador cuando son iguales.
Para entender la limitación del uso del PWM 'Unbuffered', se puede ver lo que pasa cuando se cambia el ciclo de servicio.
Como se vio antes, para generar una forma de onda PWM como la que se muestra en la figura, se necesita preparar el canal del timer para borrar el 'output compare' y se activa la salida 'toggle' en función del desbordamiento del contador. Siempre se debe usar una rutina de servicio de interrupción para manipular el ancho del pulso para evitar señales espurias, causadas por la

desincronización de escrituras en el registro de canal del timer. Las posibles elecciones son: interrupción por Comparación de Salida y por Desbordamiento del Contador.
Para los propósitos de este primer ejemplo, se ha elegido la interrupción de desbordamiento del Contador. Después, se puede ver lo que pasa cuando se usa la interrupción por desbordamiento del contador para cambiar el funcionamiento del PWM a un ciclo de servicio más pequeño. Véase que las formas de onda del ciclo de servicio mostrado en el diagrama no están dibujadas a escala.
Mientras que ocurre la petición del servicio de interrupción, en el momento de la transición de la señal de salida, tomará un tiempo para la rutina de servicio de interrupción para preparar el canal del timer para la siguiente operación de 'output compare'. Este retardo también es dependiente en si o no están en ese momento otras interrupciones en servicio. Para una aplicación que esté muy ocupada, este retardo sería casi imprevisible. Naturalmente, uno siempre debe considerar el peor caso posible.



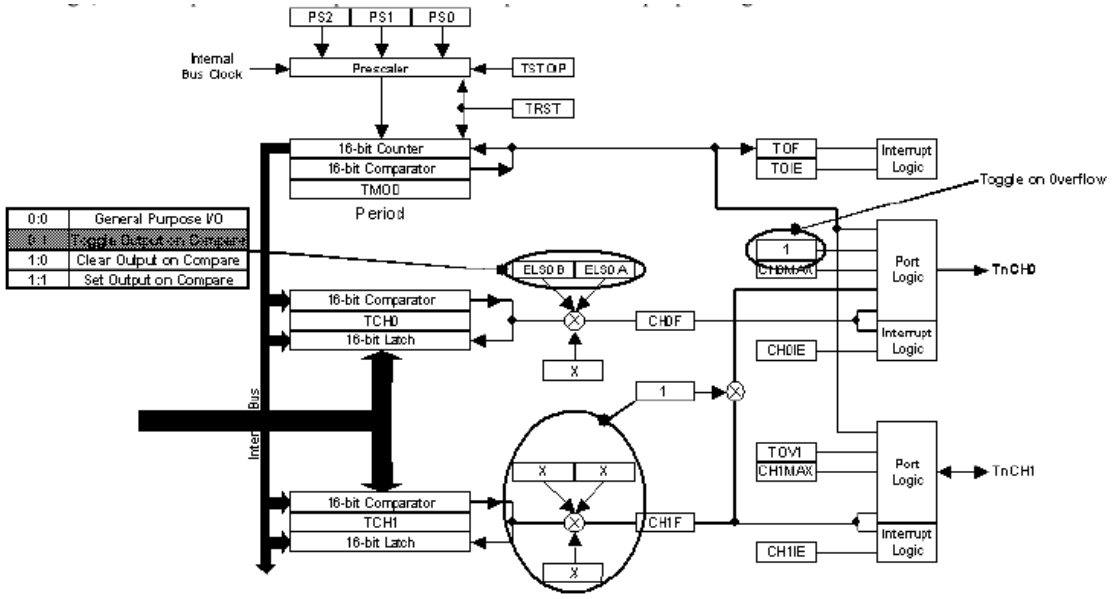
En la figura mostrada, la rutina del servicio de interrupción por desbordamiento, toma demasiado tiempo para calcular y prepararse para la siguiente transición del canal del timer. Ya que el valor en el contador ha superado el nuevo valor en el registro del canal del timer, no se ejecutará ninguna operación de 'output compare' y el estado del pin de salida no cambia.
Esto es así hasta la siguiente interrupción por desbordamiento del contador, hasta que tiempo el nivel de salida es 'toggled'; en este caso, de alto a bajo. El funcionamiento del 'output compare' todavía se configura con el nuevo valor del ciclo de servicio. Sin embargo, ya que el funcionamiento programado es borrar el pin en la comparación, el estado del pin no cambia. De hecho, solo después de la siguiente operación de desbordamiento del contador, se consigue finalmente el ciclo de servicio para el que se ha estado haciendo.
En resumen, al usar la rutina de servicio de interrupción por desbordamiento del contador para sincronizar modificaciones al ciclo de servicio del PWM, cambiando (en una aplicación específica) a un ciclo de servicio muy pequeño, causará (potencialmente) un funcionamiento indeseable. La señal experimentará 100% y después 0% ciclos de servicio antes de establecerse en el valor programado. Esto será así, cada vez que se haga un cambio a un ciclo de servicio muy pequeño. Por lo que, usando la interrupción por desbordamiento del contador quizá no es la mejor opción.

En cambio, se puede probar de usar la interrupción ‘output compare’ para sincronizar los cambios del ciclo de servicio del PWM. Como se puede ver, esta interrupción ocurrirá antes del desbordamiento del contador. Para los propósitos de este ejemplo, se cambiará el funcionamiento del PWM a un ciclo de servicio muy grande.
Así como antes, la petición del servicio de interrupción ocurre en el momento en que transiciones de las señales de salida y así tomará tiempo para la rutina del servicio de interrupción para preparar el canal del timer para la siguiente operación de ‘output compare’. Pero este retraso debe ser antes del siguiente periodo de PWM.
De hecho, muy probablemente es que este cálculo y la preparación causarán otra interrupción de ‘output compare’ para que ocurra dentro del mismo periodo de PWM. Esto no tiene ningún efecto adverso en la actividad del pin de salida y esta segunda petición de servicio de interrupción, seguramente se puede ignorar.
Como se esperaba, el desbordamiento del contador hará conmutar (‘toggle’) la salida. Y el ‘output compare’ realizará la transición correcta.
¿Pero, qué pasa cuando se reprograma el PWM durante un ciclo de servicio muy pequeño?: Se descubrirá que se tiene el mismo problema que antes. Debido a la potencia de la rutina de servicio de interrupción la respuesta se retrasa, el cálculo para el nuevo (muy pequeño) ciclo de servicio podría tomar una inaceptable cantidad de tiempo muy grande.
Y así, como se vio antes, la transición del funcionamiento PWM a través de un 100% y después del 0% del ciclo de servicio antes de establecerse en el valor programado. Esto será cierto, si se pide un cambio de un ciclo de servicio muy grande a un ciclo de servicio muy pequeño. Si la transición entre un pulso largo y un pulso corto, resulta en ninguna transición indeseable, asumiendo que es un pulso del 100% de ciclo de servicio seguido por un pulso del 0% de ciclo de servicio insertado entre ellos, analizando esta situación puede ser aceptable. Sin embargo, la clave está calculando la rutina de servicio de interrupción del ‘output compare’ con una respuesta de más largo tiempo que determina las condiciones de cuando esto ocurre. Dependiendo de la aplicación, un tiempo de respuesta más largo podría hacer este tipo de funcionamiento indeseable. Comparando las dos fuentes de interrupción, solo usando la interrupción ‘output compare’ presente un problema potencial cuando la transición del ciclo de servicio pasa de muy grande a muy pequeño. Esto es superior al usar la interrupción por desbordamiento del contador, tal como se vio en el funcionamiento anterior, cada vez que se hace una transición a un ciclo de servicio pequeño, sin tener en cuenta el valor anterior.
Investigando una posible solución a esta situación. Una vez más, se puede empezar con un ciclo de servicio muy grande. Y se escogerá la transición a un ciclo de servicio muy pequeño. Usando la interrupción ‘output compare’, tal como se vio, el proceso de calcular el nuevo valor del ciclo de servicio, puede tomar demasiado tiempo. Esta situación puede ser detectada comparando el nuevo valor con otro valor actual del contador (probablemente solo los 8-bits mas bajos) o algún otro valor predeterminado, en el peor de los casos. En lugar de usar el nuevo ciclo de servicio pequeño, se escribirá un valor intermedio durante un ciclo que se fuerza a un 50% del ciclo de servicio. Ahora el siguiente ciclo (es decir, la rutina de servicio de interrupción ‘output compare’) programa el canal del timer con el ciclo de servicio pequeño intencionado. El tiempo requerido para realizar esta tarea es de antemano en el siguiente periodo de PWM. Y el funcionamiento aquí es como se deseó. Ahora, las transiciones del 99% al 50% al 1% son probablemente aceptables en la mayoría de las aplicaciones.

PWM ‘Buffered’

Para las situaciones donde el PWM unbuffered no trabajará, se puede escoger el modo Buffered que puede quitar cualquier posibilidad de espurios en señal de salida. Se requiere el uso de dos registros del canal del timer para controlar el manejo de la señal en el pin TxCH0. El pin TxCH1

no se puede usar por el timer; sin embargo, está completamente disponible como un pin de E/S de propósito general.



Los dos canales del timer se pueden unir funcionalmente, poniendo MS0B a 1. Poniendo MS0B a 1, obligará al pin TxCH0 que sea una salida, por lo que la configuración de MS0A no le importa. Además, poniendo MS0B a 1, activa ELS0A y ELS0B para ser usado para programar el funcionamiento del pin, tal como se ha visto antes.
Ya que la actividad del pin TxCH0 está controlada a través de ELS0A y ELS0B, los otros bits de control del canal del timer, ELS1A, ELS1B y MS1A no hay que tener cuidado.
Como en el modo PWM Unbuffered, el valor guardado en el registro del canal del timer, TCH0, determina el ciclo de servicio. Y el valor en TMOD dicta el periodo de señal PWM, con el contador constantemente comparado al valor en TCH0.
Cuando el valor en el contador de 16-bits es igual a TCH0, la transición de la señal de salida programada se fuerza en señal en el pin TxCH0.
Una vez mas, como en el caso de PWM Unbuffered, se debe programar el bit TOV0 (o ‘Toggle por desbordamiento’). Poniendo TOV0 a 1 conmutará (‘toggle’) la salida cada vez que haya desbordamientos del contador de 16-bits.
Cuando el valor en el contador de 16-bits es igual a TMOD, se fuerza un desbordamiento, restablecimiento del contador a cero y con TOV=1, el pin de salida conmuta (‘toggled’).
Después, para cambiar el ciclo de servicio, el software debe programar el siguiente valor en los registros del canal del timer alternativamente. Si TCH0 está actualmente activo, entonces cualquier cambio en el ciclo de servicio PWM se escribe en TCH1 y viceversa.
Se sincronizan los cambios del ciclo de servicio con el siguiente desbordamiento del contador, determinado por TMOD. Subsecuentemente, en este ejemplo, TCH0 está actualmente activo y se ha escrito un nuevo valor en TCH1, la lógica empezará automáticamente usando TCH1 durante el siguiente ciclo de servicio.
El siguiente cambio al ciclo de servicio requerirá una escritura en TCH0. el software del usuario necesitará guardar la traza de lo que fue escrito anteriormente en el registro del canal del timer, para determinar que canal será escrito a continuación. Pero, como que ahora se sincronizan automáticamente los cambios del ciclo de servicio con el timer, este software ya no necesita residir dentro de una rutina de servicio de interrupción del timer.

Cuando el valor en el contador de 16-bits es igual a TCH1, la transición de la señal de salida programada se fuerza en el pin la señal TxCH0. se generará el PWM deseado hasta un nuevo ciclo de servicio, que es programado en los registros TCH1 o TCH0.

APÉNDICE C

PROGRAMACIÓN

APÉNDICE C
PROGRAMACIÓN

C.1 CONSIDERACIONES DE PROGRAMACIÓN

Existen muchas cuestiones relativas en parte al programa específico que se esté desarrollando y en parte al sistema en que se implementa dicho programa, que deben ser analizadas en pro del mejoramiento de su funcionamiento.

En términos más específicos, se deben discernir cuestiones tales como uso y distribución de los tiempos de programa, diseño de rutinas (secuencia y ubicación de las mismas), manejo de los registros (propios del microcontrolador usado y creados por programador). Éstas serán desarrolladas a continuación.

USO Y DISTRIBUCIÓN DE TIEMPOS DEL PROGRAMA

Dependiendo fundamentalmente de las características del sistema que se desea desarrollar, es necesario definir cuantitativamente los tiempos en los que el sistema se encuentra ocupado, o bien ocioso. Los primeros son de mayor importancia en la respuesta del sistema a estímulos externos, por ejemplo a pedidos de comunicación desde dispositivos externos; mientras que los tiempos de ocio definen en mayor medida la eficiencia en el comportamiento del sistema.

En el caso particular del sistema desarrollado, el mayor hincapié se realizó sobre los tiempos en los que el sistema se encuentra ocupado. En este sentido, es importante que el sistema se encuentre disponible en los tiempos que el usuario deba emplearlo –ya sea transmisor o receptor- . Por esto no es conveniente entre otras cosas, desarrollar rutinas o gran cantidad de instrucciones dentro de las interrupciones, dado que dentro de una interrupción, el sistema se haya prácticamente “incomunicado” con el exterior. Así, la configuración -para el caso en que sea automática- del código del transmisor debe emplear la menor cantidad de tiempo posible para posibilitar una comunicación en cualquier momento en el que el usuario lo desee. Otros ejemplos relativos a la distribución de tiempos ya han sido desarrollados en el receptor, tales como la inicialización del display LCD.

Finalmente cabe señalar que es muy probable que en la búsqueda de minimizar los tiempos ocupados se choque con su contraparte de aumentar los tiempos ociosos y así, la ineficiencia del sistema. Prueba de ello son el sistema aquí desarrollado, en el que el sistema se encuentra en estado ocioso la mayor parte del tiempo, para asimismo, hallarse disponible a los requerimientos del usuario.

DISEÑO DE RUTINAS

Las rutinas son la base en el diseño modular de todo programa. Y lo más probable es que éstas se relacionen la una con la otra de diversas formas, ya sea por uso de registros comunes, flags determinados, sensado de puertos, otra rutina, etc.
Si bien las distintas rutinas se hallan en general interconectadas, es necesario definirlas cada una por separado (la esencia del diseño modular) para simplificar luego su posterior simulación y análisis en conjunto. La primer rutina que se debe definir siempre es la de inicialización, apenas el sistema es encendido. Si bien la rutina de inicialización puede ubicarse en cualquier sector de la memoria ROM (o FLASH en nuestro caso), siempre es recomendable hacerlo al comienzo de ésta para una mejor organización. También es aconsejable que en esta primer rutina no se incluyan subrutinas o funciones específicas del programa.
Las otras rutinas que aparecen en casi cualquier programa son las rutinas de interrupción que como se vio en esta misma sección no deben ser demasiado extensos. Para ello, una de las alternativas es el uso de registros/ bits llamados flags que se "enciendan" (coloquen su valor lógico en 1) o que se "apaguen" (coloquen su valor lógico en 0) dentro de estas interrupciones, para luego ser sensadas en alguna otra parte del programa, fuera de las interrupciones.
Finalmente en cuanto a la disposición de las distintas rutinas en memoria, es recomendable para el funcionamiento del programa, la mayor proximidad posible de rutinas y subrutinas que se llamen la una a la otra, elemento que se torna casi obligatorio en el caso de uso de saltos de desplazamiento que tienen un determinado rango de posiciones en las que puede actuar.

MANEJO DE LOS REGISTROS

Los registros -ya sea tanto los definidos por el microcontrolador empleado como los creados por el programador- son en general de tipo dinámico, es decir no presentan un valor fijo o al menos no a lo largo de todo el programa. En el caso específico del programa del sistema desarrollado se hizo uso de registros de puertos entrada/salida (direccionamiento y datos), registros relativos a distintos módulos (registros del módulo TIM, del módulo ADC, del módulo COP, etc.), así como se definieron otros como contadores, flags, registros intermedios, entre otros.
Entre las consideraciones que se deben hacer en el manejo de los registros, está el uso de registros "comodines" o intermedios. Estos son utilizados para almacenar valores intermedios por ejemplo cuando deben realizarse máscaras con operaciones lógicas en algún registro, particularmente en la escritura de puertos o registros propios del microcontrolador, en los que no se debe trabajar directamente para evitar posibles complicaciones . Relacionado con esto, en la configuración de los puertos como salidas, también se aconseja primero colocar el valor lógico deseado en el puerto correspondiente y recién después direccionarlo como salida. Esto previene comportamientos erráticos como el fenómeno de 'glitching' (valores espúreos en la salida)

C.2 CÓDIGO FUENTE
C.2.1 TRANSMISOR

```
*******************************************************************************
------------          E T I Q U E T A S              ---------------
*******************************************************************************

$Include 'jl3regs.inc'
flash_start_jk1   equ $f600
reset_vec         equ $fffe
irq_int           equ $fffa
adc_int           equ $ffde
contador          equ $0080
contador2         equ $0081
flagsdigito       equ $0097
newoffset         equ $0098
newoffset2        equ $0099
conteo1           equ $00a0
conteo2           equ $00a1
cuenta66mili      equ $00a2
contdig1          equ $00a5
contdig2          equ $00a6
contdig3          equ $00ab
contdig4          equ $00a8
nroconv           equ $00ad
oc_vec            equ $fff6
toverflow         equ $fff2
IMASK1            equ 1
ACK1              equ 2
MODE1             equ 0
KEYF              equ 3
PTDPU6            equ 0
BCFE              equ 7
progdigit         equ $00a3
flagover          equ 7
fin_conv          equ 6
cuenta100mili     equ $00a4
cuenta300mili     equ $00b0
emergencia        equ $00b1
temp_adc          equ $00a9
temp_digit        equ $00aa
tono0             equ $0082
tono1             equ $0083
tono2             equ $0084
tono3             equ $0085
tono4             equ $0086
tono5             equ $0087
```

```
tono6           equ $0088
tono7           equ $0089
tono8           equ $0090
tono9           equ $0091
tonoa           equ $0092
tonob           equ $0093
tonoc           equ $0094
tonod           equ $0095
tonoe           equ $0096



*****************************************************************************
----------      CONFIGURACIÓN GENERAL              ----------
*****************************************************************************

.org flash_start_jk1

start:
    clrx
    lda #$01  ;cargo uno en el acc
    sta $001f ;deshabilito COP en el config register
    lda #$30  ;cargo $30 en el acc
    sta $0020 ;para habilitar el TSTOP Y TRST del TSC
    lda #$a1
    sta $0023 ;cargo la parte alta del TIM counter modulo reg
    lda #$22
    sta $0024 ;cargo la parte baja de TIM counter modulo reg (41250 dec, corresp.33 miliseg.)
    lda #$d0
    sta ddrd
    lda #$80
    sta ptd
    sta ddrb
    bclr 7,ptb
    bset IMASK1,INTSCR ; Seteo la máscara para que no interrumpa
    lda #$80
    sta bfcr
    mov #!10,nroconv
    clr emergencia
    clr flagsdigito
    clr cuenta66mili
    clr cuenta100mili
    clr cuenta300mili
    clr contador
    clr contador2
    clr conteo1
    clr contdig1
    clr contdig2
    clr contdig3
```

```
    clr contdig4
    clr progdigit
    clr tono0
    clr tono1
    clr tono2
    clr tono3
    clr tono4
    clr tono5
    clr tono6
    clr tono7
    clr tono8
    clr tono9
    clr tonoa
    clr tonob
    clr tonoc
    clr tonod
    clr tonoe
    bset flagover,progdigit
    bset ACK1,INTSCR ;limpio la llamada de int para evitar falsas int.
    bclr IMASK1,INTSCR ;deshabilito la máscara
    cli

*************************************************************************
----------CONFIGURACIÓN / LECTURA DEL CÓDIGO DE TRANSMISIÓN----------
*************************************************************************

      brset 0,ptb,no_prog    ;usamos ptb0 como indicador de donde
      bclr flagover,progdigit ;debe tomar el codigo
      bset 1,emergencia
      brset 6,ptb,mili33
      jsr adc_cod_conf       ;NUEVA OPCION!!!!!!!!!!!!!
      jsr adc_cod_conf       ; USO DEL CONV ADC CON INT
      jsr adc_cod_conf       ;PARA DECIDIR EL CÓDIGO
      jsr adc_cod_conf
      mov #$1f,ADSCR
      bset flagover,progdigit
      bclr 1,emergencia
      clr conteo1
mili33    brset 1,emergencia,mili33
        brset 2,ptd,mili33                ;0---lo toma de los dip-switch
        bset 0,emergencia
        bset 1,emergencia
emer    jmp rutina                        ;1----lo toma de memoria
no_prog  MOV #$82,contador
      MOV #$34,contador2
      jmp mili33


*************************************************************************
----------CONFIGURACIÓN INGRESO DEL CÓDIGO POR CONVERSORES A/D----------
```

```
*************************************************************************


adc_cod_conf    inc conteo1
sigueconv       ldx conteo1
                lda tabla_adc,x
                sta adscr
nada_adc        brset 7,ADSCR,dec_cod
                bra nada_adc
dec_cod         lda adr
                sta temp_adc
                inc progdigit
                jsr adc_cod_dec
                dec nroconv
                lda nroconv
                bne sigueconv
                jsr digdec
                jsr clrdec
                lda progdigit
                cbeqa #$0a,rutinaprog_dig1
                cbeqa #$14,rutinaprog_dig2
                cbeqa #$1e,rutinaprog_dig3
                cbeqa #$28,rutinaprog_dig4
                bset flagover,progdigit
                rts

adc_cod_dec     lda temp_adc
                lsra
                lsra
                lsra
                lsra
                jsr decide_dig

                rts
rutinaprog_dig1 lda temp_digit
                sta contdig1
                bclr fin_conv,progdigit
                 rts
rutinaprog_dig2         lda temp_digit
                        lsla
                        lsla
                        lsla
                        lsla
                        sta contdig2
                        lda contdig1
                        ora contdig2
                        sta contador
                        bclr fin_conv,progdigit
                        rts
rutinaprog_dig3         lda temp_digit
```

```
                sta contdig3
                bclr fin_conv,progdigit
                rts
rutinaprog_dig4     lda temp_digit
                lsla
                lsla
                lsla
                lsla
                sta contdig4
                lda contdig3
                ora contdig4
                sta contador2
                bclr fin_conv,progdigit
                rts
decide_dig      cbeqa #0,rut0
            dbnza rutsig1
            inc tono1
            rts
rut0          jmp rutsig0
rutsig1         dbnza rutsig2
            inc tono2
            rts
rutsig2         dbnza rutsig3
            inc tono3
            rts
rutsig3   dbnza rutsig4
            inc tono4
            rts
rutsig4   dbnza rutsig5
            inc tono5
            rts
rutsig5   dbnza rutsig6
            inc tono6
            rts
rutsig6   dbnza rutsig7
            inc tono7
            rts
rutsig7   dbnza rutsig8
            inc tono8
            rts
rutsig8   dbnza rutsig9
            inc tono9
            rts
rutsig9   dbnza rutsiga
            inc tonoa
            rts
rutsiga   dbnza rutsigb
            inc tonob
            rts
rutsigb   dbnza rutsigc
```

```
            inc tonoc
            rts
rutsigc   dbnza rutsigd
            inc tonod
            rts
rutsigd       inc tonoe
            rts
rutsig0       inc tono0
            rts

digdec        ;USAMOS UNA RUTINA OPTIMIZADA PARA HALLAR EL REGISTRO
            ; CON EL NRO MAYOR
            lda #$a
            sta nroconv
-           clrx
            lda tono0
            cmp tono1
            blo siguetono1
sig1          cmp tono2
            blo siguetono2
sig2          cmp tono3
            blo siguetono3
sig3          cmp tono4
            blo siguetono4
sig4          cmp tono5
            blo siguetono5
sig5          cmp tono6
            blo siguetono6
sig6          cmp tono7
            blo siguetono7
sig7          cmp tono8
            blo siguetono8
sig8          cmp tono9
            blo siguetono9
sig9          cmp tonoa
            blo siguetonoa
siga          cmp tonob
            blo siguetonob
sigb          cmp tonoc
            blo siguetonoc
sigc          cmp tonod
            blo siguetonod
sigd          cmp tonoe
            bhi decide
            ldx #$e
decide        stx temp_digit
            rts
siguetono1      lda tono1
            ldx #$1
            bra sig1
```

```
siguetono2      lda tono2
                ldx #$2
                bra sig2
siguetono3      lda tono3
                ldx #$3
                bra sig3
siguetono4      lda tono4
                ldx #$4
                bra sig4
siguetono5      lda tono5
                ldx #$5
                bra sig5
siguetono6      lda tono6
                ldx #$6
                bra sig6
siguetono7      lda tono7
                ldx #$7
                bra sig7
siguetono8      lda tono8
                ldx #$8
                bra sig8
siguetono9      lda tono9
                ldx #$9
                bra sig9
siguetonoa      lda tonoa
                ldx #$a
                bra siga
siguetonob      lda tonob
                ldx #$b
                bra sigb
siguetonoc      lda tonoc
                ldx #$c
                bra sigc
siguetonod      lda tonod
                ldx #$d
                bra sigd

clrdec          clr tono0
                clr tono1
                clr tono2
                clr tono3
                clr tono4
                clr tono5
                clr tono6
                clr tono7
                clr tono8
                clr tono9
                clr tonoa
                clr tonob
                clr tonoc
```

```
            clr tonod
            clr tonoe
            rts


tabla_adc FCB  $00,$03,$04,$01,$02

tabla_code FCB 0,1,2,3,4,5,6,7,8,9,$a,
        FCB $b,$c,$d,$e,$f

*****************************************************************************
----CONFIGURACIÓN INGRESO DE CÓDIGO POR DIP-SWITCH / TRASMISIÓN     ------
*****************************************************************************
rutina:
   bset IMASK1,INTSCR
   brset flagover,progdigit,envio1
   inc progdigit
   lda progdigit
   cbeqa #$1,rutinaprog_dig1a
   cbeqa #$2,rutinaprog_dig2a
   cbeqa #$3,rutinaprog_dig1a
   cbeqa #$4,rutinaprog_dig2a
   bset flagover,progdigit
   bclr IMASK1,INTSCR
   rti
envio1 jmp envio
rutinaprog_dig1a        lda ptb
                  lsla
                  lsla
                  lsla
                 brset 1,progdigit,progdig3
                  sta contdig1
                  jsr delay300mili
                  bclr IMASK1,INTSCR
                 rti
progdig3               sta contdig3
                  jsr delay300mili
                  bclr IMASK1,INTSCR
                  rti
rutinaprog_dig2a        lda ptb
                  lsra
                  brclr 1,progdigit,progdig2
                  sta contdig2
                  lda #$0f
                  ora contdig1
                  sta contdig1
                  lda #$f0
                  ora contdig2
                  sta contdig2
                  and contdig1
```

```
                sta contador
                 jsr delay300mili
                 bclr IMASK1,INTSCR
                 rti
progdig2              sta contdig4
                 lda #$0f
                 ora contdig3
                sta contdig3
                 lda #$f0
                ora contdig4
                 sta contdig4
                 and contdig3
                 sta contador2
                jsr delay300mili
                 bset flagover,progdigit
                 bclr IMASK1,INTSCR
                 bclr 1,emergencia
                 rti

*****************************************************************************
------------   RETARDO ANTIRREBOTE                    ---------------
*****************************************************************************
delay300mili          mov #10,cuenta300mili
siguedec31            ldx #!100
siguedec21            lda #$ff
siguedec1            dbnza siguedec1
                 dbnzx siguedec21
                 dbnz cuenta300mili,siguedec31
                 bset ACK1,INTSCR
                 rts
*****************************************************************************
------------     CONFIGURACIÓN DE LA TRANSMISIÓN          ---------------
*****************************************************************************
envio      lda #$80 ;activamos el flag de int como salida de PTB7
         sta ptb   ; hacia el tx para evitar pérdidas de envios
         bset 6,ptd
         jsr delay300mili
         brclr 0,emergencia,normal_mode
*****************************************************************************
------------   CONFIGURACIÓN MODO EMERGENCIA            ---------------
*****************************************************************************
         lda #$e
         bra rutgral
         lda #$90
         sta ptd
*****************************************************************************
------------CONFIGURACIÓN MODO NORMAL / TRANSMISIÓN PRIMER DÍGITO---------
*****************************************************************************
normal_mode bset 0,flagsdigito ;activamos el flag de que pone el primer digito DEBERIA IR
DESPUES DE HABERLO HECHO!!!!!!!!!!
```

```
        lda #$f0  ;hacemos una máscara para sacar los 4 primeros bits
        and contador ;del primer contador
        nsa ;desplazo los 4 bits MSB al lugar de los 4 LSB para ver a la rutina de que frec van
        bra rutgral
*****************************************************************************
----------  CONFIGURACIÓN TRANSMISIÓN SEGUNDO DÍGITO  ----------
*****************************************************************************

rutina2  bset 1,flagsdigito
       lda #$0f ;hacemos máscara para sacar los segundos 4 bits
       and contador;del primer contador
       bra rutgral
*****************************************************************************
----------  CONFIGURACIÓN TRANSMISIÓN TERCER DÍGITO  ----------
*****************************************************************************

rutina3 bset 2,flagsdigito
      lda #$f0  ;hacemos una máscara para sacar los 4 primeros bits
      and contador2 ;del segundo contador
      nsa ;desplazo los 4 bits MSB al lugar de los 4 LSB para ver a la rutina de que frec van
      bra rutgral

*****************************************************************************
----------  CONFIGURACIÓN TRANSMISIÓN CUARTO DÍGITO  ----------
*****************************************************************************
rutina4 bset 3,flagsdigito
      lda #$0f ;hacemos máscara para sacar los segundos 4 bits
      and contador2;del segundo contador
      bra rutgral

*****************************************************************************
----------  RUTINA GENERAL DE TRANSMISIÓN          ----------
*****************************************************************************

rutgral
      cbeqa #0,frq0  ;si da cero va a la rutina de frecuencia cero
      cbeqa #1,frq1 ;si da uno va a la rutina de frecuencia uno
      cbeqa #2,frq2;si da dos va a la rutina de frecuencia dos
      cbeqa #3,frq3;si da tres va a la rutina de frecuencia tres
      cbeqa #4,frq4;si da cuatro va a la rutina de frecuencia cuatro
      cbeqa #5,frq5;si da cinco va a la rutina de frecuencia cinco
      cbeqa #6,frq6;si da seis va a la rutina de frecuencia seis
      cbeqa #7,frq7;si da siete va a la rutina de frecuencia siete
      cbeqa #8,frq8;si da ocho va a la rutina de frecuencia ocho
      cbeqa #9,frq9;si da nueve va a la rutina de frecuencia nueve
      cbeqa #$a,frqa;si da diez va a la rutina de frecuencia diez
      cbeqa #$b,frqb;si da once va a la rutina de frecuencia once
      cbeqa #$c,frqc ;si da doce va a la rutina de frecuencia doce
      cbeqa #$d,frqd;si da trece va a la rutina de frecuencia trece
      cbeqa #$e,frqe; si da catorce va la rutina de frec.catorce
```

```
frq0    ldhx #!631;cargamos en el registro índice 315 dec 304
        bra pulso
frq1    ldhx #!1112;cargamos en el registro índice 556 dec 535
        bra pulso
frq2    ldhx #!1044;cargamos en el registro índice 522 dec 503
        bra pulso
frq3    ldhx #!980;cargamos en el registro índice 490 dec 472
        bra pulso
frq4    ldhx #!920;cargamos en el registro índice 460 dec 443
        bra pulso
frq5    ldhx #!864;cargamos en el registro índice 432 dec 416
        bra pulso
frq6    ldhx #!812;cargamos en el registro índice 406 dec 391
        bra pulso
frq7    ldhx #!763;cargamos en el registro índice 381 dec 367
        bra pulso
frq8    ldhx #!716;cargamos en el registro índice 358 dec 344
        bra pulso
frq9    ldhx #!672;cargamos en el registro índice 336 dec 323
        bra pulso
frqa    ldhx #!521;cargamos en el registro índice 260 dec 251
        bra pulso
frqb    ldhx #!1344;cargamos en el registro índice 672 dec 647
        bra pulso
frqc    ldhx #!557;cargamos en el registro índice 278 dec 268
        bra pulso
frqd    ldhx #!1261;cargamos en el registro índice 630 dec 607
        bra pulso
frqe    ldhx #!593;cargamos en el registro índice 296 dec 285
        bra pulso


*****************************************************************************
------------   CONFIGURACIÓN MODO OUTPUT COMPARE          ---------------
*****************************************************************************

pulso  sthx tch0h;posición 26
       sthx newoffset
       sthx conteo1
       lda #$54
       sta tsc0 ;posicion $0025 configuro el tsc0 como output compare con toggle
       ;lda #$10
       ;sta ddrd ;pongo el ptd4 como salida (pin de salida del oc)SE DEBE PONER ANTES!!!!!!!!
       brset 0,emergencia,mode_E
       lda #$50
       sta tsc  ;dirección $0020 reseteo el timer
       rti
mode_E bclr 0,emergencia
       lda #$50
```

```
      sta tsc  ;dirección $0020 reseteo el timer
      jmp mili33

OC_Int   BCLR  7,TSC0
        RSP
        ;lda #$20
        ;sta tsc
        lda conteo2
        adc newoffset2
        sta newoffset2
        bcc sumacont
        inc newoffset
sumacont lda conteo1
        add newoffset
        sta newoffset
        sta tch0h
        lda newoffset2
        sta tch0l
        CLI
        lda #$40
        sta tsc
        jmp mili33

*****************************************************************************
------------              BASE DE TIEMPO              --------------
*****************************************************************************
sigue:
        bclr 7,tsc
        lda #$80
        sta ptd
        inc cuenta66mili
        lda cuenta66mili
        cbeqa #3,continua
        rti
continua clr cuenta66mili
        brset 0,flagsdigito,sigue1
        jmp normal_mode
sigue1   brset 1,flagsdigito,sigue2 ;si el flag de pasar por el segundo digito está
        jmp rutina2;activado va a preguntar por el siguiente, sino va a mandarlo
sigue2   brset 2,flagsdigito,sigue3
        jmp rutina3
sigue3   brset 3,flagsdigito,final
        jmp rutina4
final    lda #$30
        sta tsc
        clr flagsdigito
        bclr 1,emergencia
        bclr 7,ptb ;borramos el flag de llamada de int . al tx
        bclr 6,ptd
        bset ACK1,INTSCR
```

```
        bclr IMASK1,INTSCR ;limpio el registro de int por teclado
         rti

.org irq_int
   dw rutina
.org reset_vec
   dw start
.org oc_vec
   dw oc_int
.org toverflow
   dw sigue
```

C.2.2 RECEPTOR

```
*****************************************************************************
------------          E T I Q U E T A S              ---------------
*****************************************************************************

$Include 'jl3regs.inc'
flash_start_jk3    equ $ec00
reset_vec          equ $fffe
irq_int            equ $fffa
contador           equ $0080
contador2          equ $0081
flagsdigito        equ $0097
newoffset          equ $0098
newoffset2         equ $0099
conteo1            equ $00a0
conteo2            equ $00a1
cuenta66mili       equ $00a2
ic_vec             equ $fff6
toverflow          equ $fff2
IMASK1             equ 1
ACK1               equ 2
MODE1              equ 0
KEYF               equ 3
PTDPU6             equ 0
BCFE               equ 7
flagchequeo        equ $00a3
nrodigito          equ $00a4
tono0              equ $0082
tono1              equ $0083
tono2              equ $0084
tono3              equ $0085
tono4              equ $0086
tono5              equ $0087
tono6              equ $0088
tono7              equ $0089
tono8              equ $0090
tono9              equ $0091
tonoa              equ $0092
tonob              equ $0093
tonoc              equ $0094
```

```
tonod             equ $0095
tonoe             equ $0096
digito1           equ $00a5
digito2           equ $00a6
digito3           equ $00a7
digito4           equ $00a8
digito0           equ $00bd
digitof1          equ $00be
digitof2          equ $00bf
digitof3          equ $00c0
digitof4          equ $00c1
digitof0          equ $00c2
digitof5          equ $00c3
digitof6          equ $00c4
digitof7          equ $00c5
digitof8          equ $00c6
digitof9          equ $00c7
digitofa          equ $00c8
digitofb          equ $00c9
digitofc          equ $00ca
digitofd          equ $00cb
digitofe          equ $00cc
temp              equ $00a9
pulsador          equ $00aa
puls_enable       equ $00ab
flag_ruido        equ $00d0
lcdtemp    equ $00ac
lcdtemp2   equ $00ad
eeadr      equ $00ae
eeadrh     equ $00af
lcd_reg    equ $00b0
char1      equ $00b1
char2      equ $00b2
char3      equ $00b3
char4      equ $00b4
char5      equ $00b5
char6      equ $00b6
char7      equ $00b7
char8      equ $00b8
estado     equ $00b9
lcdcont    equ $00ba
cuenta5mili equ $00bb
nrotxfreez equ $00bc
cuenta5seg equ $00cd
contpulsreset equ $00ce
tx          equ $00cf
fuera_rango   equ $00d1
cant_tonos    equ $00d2
cuenta300mili equ $00d3
do_display equ 0
```

```
flag_inic_lcd equ 1
tx_previa     equ 2
line          equ 0
wmessage      equ 3
read          equ 4



*****************************************************************************
------------        CONFIGURACIÓN GENERAL                 ----------------
*****************************************************************************
.org flash_start_jk3

start:
      clrx
      lda #$01  ;cargo uno en el acc
      sta $001f ;deshabilito COP en el config register
      lda #$30  ;cargo $30 en el acc
      sta $0020 ;para habilitar el TSTOP Y TRST del TSC
      lda #$a1
      sta $0023 ;cargo la parte alta del TIM counter modulo reg
      lda #$22
      sta $0024 ;cargo la parte baja de TIM counter modulo reg (41250 dec, corresp.33 miliseg.)
      lda #$c0
      sta ddrd
      clra
      sta ptd
      clr flagsdigito
      clr newoffset
      clr newoffset2
      clr nrotxfreez
      clr cuenta66mili
      clr cuenta5mili
      clr cuenta5seg
      clr contpulsreset
      clr flagchequeo
      clr nrodigito
      clr digito0
      clr digito1
      clr digito2
      clr digito3
      clr digito4
      clr pulsador
      clr puls_enable
      clr tx
      clr temp
      clr fuera_rango
      clr cant_tonos
      clr cuenta300mili
      clr tono0
```

```
        clr tono1
        clr tono2
        clr tono3
        clr tono4
        clr tono5
        clr tono6
        clr tono7
        clr tono8
        clr tono9
        clr tonoa
        clr tonob
        clr tonoc
        clr tonod
        clr tonoe
        clr lcd_reg
        clr estado
        clr lcdtemp
        clr lcdtemp2
        clr flag_ruido
        mov #' ',char1
        mov #' ',char2
        mov #'H',char3
        mov #'O',char4
        mov #'L',char5
        mov #'A',char6
        mov #' ',char7
        mov #' ',char8
        cli

*****************************************************************************
----PROGRAMA INICIALIZACIÓN DISPLAY /MENÚ PRINCIPAL PRE-RECEPCIÓN -------
*****************************************************************************

        jsr delay300mili
        jsr inic_display
        jsr presentacion
test     brset 0,puls_enable,reset_display;OPCION DE RESET DE DISPLAY
        brset 2,puls_enable,sigtest
        brclr 1,puls_enable,sigtest
        lda #$50
        sta tsc
        bset 2,puls_enable
sigtest  brset 4,ptd,transmite
        bra test
*****************************************************************************
-----------   RECONFIGURACIÓN PARA LA RECEPCIÓN            -----------
*****************************************************************************

transmite lda #$50
        sta tsc
```

```
        LDA #$48
        STA TSC0 ;(ENTRADA POR PTD4)
        bset 0,tx
*****************************************************************************
------------  MENÚ PRINCIPAL DE CEPCIÓN            -----------
*****************************************************************************
wait     brset 0,flagchequeo,go_check
        brset do_display,lcd_reg,lcd_routine1
        brclr 0,flag_ruido,wait
        bclr 0,flag_ruido
        jmp test

*****************************************************************************
------------  RESET DISPLAY            -----------
*****************************************************************************

reset_display
             lda #$01;escribo 00000001 en el display LCD CLEAR DISPLAY              jsr
command_write
             jsr command_write
             bclr tx_previa,lcd_reg
             bclr 0,puls_enable
             jmp test
*****************************************************************************
------------  RUTINA DISPLAY  GENERAL            -----------
*****************************************************************************

lcd_routine1
         bclr do_display,lcd_reg
         brset 4,puls_enable,freezing;OPCION DE CONGELADO DE DISPLAY
repite     jsr config_char
         jsr message
         lda nrotxfreez
         beq finlcd
         jsr delay5seg
         bra repite
finlcd     bclr do_display,lcd_reg
         bclr 6,ptd
         jmp test
freezing  jmp freezing1

*****************************************************************************
------------  PRESENTACIÓN DISPLAY INICIALIZADO            -----------
*****************************************************************************
presentacion    jsr message
             bset line,estado
             mov #'C',char1
             mov #'E',char2
             mov #'N',char3
             mov #'T',char4
```

```
            mov #'R',char5
            mov #'A',char6
            mov #'L',char7
            mov #' ',char8
            jsr message
            bclr line,estado
            rts
*******************************************************************************
------------  RETARDO PRE-INICIALIZACION DISPLAY            -----------
*******************************************************************************
delay300mili        mov #10,cuenta300mili
siguedec300         ldx #!100
siguedec200         lda #$ff
siguedec100         dbnza siguedec100
                    dbnzx siguedec200
                    dbnz cuenta300mili,siguedec300
                    bset ACK1,INTSCR
                    rts

test1       jmp test
go_check    jmp go_check1
*******************************************************************************
------------  RUTINA DE DETECCIÓN           -----------
*******************************************************************************
sigue4      lda conteo2
            cmp #$5a
            blo sigue4_1
            cmp #$d1
            blo sigue4_2
            cmp #$fe
            bhi sigue4_3
            bra fin_sigue2
sigue4_1    cmp #$0c
            blo fin_sigue2
            inc tono4
            bra fin_sigue2
sigue4_2    cmp #$82
            blo fin_sigue2
            inc tono3
            bra fin_sigue2
sigue4_3    inc tono2
            bra fin_sigue2
sigue5      lda conteo2
            cmp #$56
            blo sigue5_1
            cmp #$e0
            blo sigue5_2
            bra fin_sigue2
sigue5_1    inc tono2
            bra fin_sigue2
```

```
sigue5_2      cmp #$82
              blo fin_sigue2
              inc tono1
              bra fin_sigue2
sigue6        lda conteo2
              cmp #$a7
              blo fin_sigue2
              inc tonod
fin6          bra fin_sigue2
sigue7        lda conteo2
              cmp #$0e
              bhi sigue7_1
              inc tonod
sigue7_1      cmp #$b8
              bhi fin_sigue2
              cmp #$48
              blo fin_sigue2
              inc tonob
              bra fin_sigue2
sigue8        lda conteo2
              cmp #$fb
              blo fin_sigue2
              inc tonoa
              bra fin_sigue2
fin_sigue2    bclr 0,flagchequeo
              jmp wait
              rts




go_check1   jmp check
wait2       brset 0,flagchequeo,go_check1
            brset do_display,lcd_reg,lcd_routine
            brclr 0,flag_ruido,wait2
            bclr 0,flag_ruido
            jmp test

lcd_routine
             bclr do_display,lcd_reg
             brset 4,puls_enable,freezing1
repite1      jsr config_char
             jsr message
             lda nrotxfreez
             beq finlcd1
             jsr delay5seg
             bra repite1
finlcd1      bclr do_display,lcd_reg
             jmp test1
go_sigue4  jmp sigue4
```

```
go_sigue5   jmp sigue5
go_sigue6   jmp sigue6
go_sigue7   jmp sigue7
go_sigue8   jmp sigue8

freezing1  inc nrotxfreez
          lda nrotxfreez
          cbeqa #1,freez11
          cbeqa #2,freez21
          bclr 4,puls_enable
          jmp repite1
freez11       mov digito0,digitof0
           mov digito1,digitof1  ;SE GUARDAN LOS DIGITOS CONGELADOS
          mov digito2,digitof2  ;EN REGISTROS TIPO "BUFFER" PARA
          mov digito3,digitof3  ;NO PERDERLOS SI HAY UN NUEVO ENVIO
          mov digito4,digitof4
          bclr 6,ptd
          jmp test1
freez21       mov digito0,digitof5
           mov digito1,digitof6  ;SE GUARDAN LOS DIGITOS CONGELADOS
          mov digito2,digitof7  ;EN REGISTROS TIPO "BUFFER" PARA
          mov digito3,digitof8  ;NO PERDERLOS SI HAY UN NUEVO ENVIO
          mov digito4,digitof9
          bclr 6,ptd
          jmp test1
rutina_fuera_rango      inc fuera_rango
                    bclr 0,flagchequeo
                    jmp wait2

check         lda conteo1
             cmp #$3
             blo rutina_fuera_rango
             cmp #$a
             bhi rutina_fuera_rango
             cbeqa #$4,sigue1
             cbeqa #$5,sigue2
             cbeqa #$6,sigue3
             cbeqa #$7,go_sigue4
             cbeqa #$8,go_sigue5
             cbeqa #$9,go_sigue6
             cbeqa #$a,go_sigue7
             cbeqa #$3,go_sigue8
             lda #$80
             sta ptd
             bclr 0,flagchequeo
             jmp wait2
sigue1        lda conteo2
             cmp #$26
             blo sigue1_1
             cmp #$70
```

```
            blo sigue1_2
            cmp #$bb
            blo sigue1_3
            cmp #$d4
            bhi sigue1_4
fin_sigue1    bclr 0,flagchequeo
            jmp wait2
*****************************************************************************
------------   RUTINA DE DETECCIÓN - CONTINUACIÓN            -----------
*****************************************************************************

sigue1_1      inc tonoa
            bra fin_sigue1
sigue1_2      cmp #$44
            blo fin_sigue1
            inc tonoc
            bra fin_sigue1
sigue1_3      cmp #$89
            blo fin_sigue1
            inc tonoe
            bra fin_sigue1
sigue1_4      inc tono0
            bra fin_sigue1
sigue2        lda conteo2
            cmp #$07
            blo sigue2_1
            cmp #$5c
            blo sigue2_2
            cmp #$b4
            blo sigue2_3
            cmp #$d7
            bhi sigue2_4
            bra fin_sigue1
sigue2_1      inc tono0
            bra fin_sigue1
sigue2_2      cmp #$24
            blo fin_sigue1
            inc tono9
            bra fin_sigue1
sigue2_3      cmp #$79
            blo fin_sigue1
            inc tono8
            bra fin_sigue1
sigue2_4      inc tono7
            bra fin_sigue1
sigue3        lda conteo2
            cmp #$15
            blo sigue3_1
            cmp #$76
            blo sigue3_2
```

```
          cmp #$e4
          blo sigue3_3
          bra fin_sigue1
sigue3_1      inc tono7
          bra fin_sigue1
sigue3_2      cmp #$38
          blo fin_sigue1
          inc tono6
          bra fin_sigue1
sigue3_3      cmp #$9d
          blo fin_sigue1
          inc tono5
          bra fin_sigue1



*****************************************************************************
----------   RETARDO MODO FREEZING            -----------
*****************************************************************************

delay5seg             mov #!10,cuenta5seg
siguedec3_5seg        ldx #!200  ;base de tiempo de 1mseg
siguedec2_5seg        lda #!171
siguedec_5seg         dbnza siguedec_5seg
                  dbnzx siguedec2_5seg
                  dbnz cuenta5seg,siguedec3_5seg
                  rts

*****************************************************************************
----------   RECONFIGURACIÓN MODO INPUT CAPTURE            -----------
*****************************************************************************
ic_int:       bclr 7,tsc0
          lda tch0h
          sub newoffset
          sta conteo1
          lda tch0h
          sta newoffset
          lda tch0l
          sub newoffset2
          bcc no_overflow
          dec conteo1
no_overflow   sta conteo2
          lda tch0l
          sta newoffset2
          inc nro_flanco
          lda nro_flanco
          cmp #2
          blo finic
          cmp #3
          blo compara
```

```
            lda siguiente
            cmp conteo1
            blo compara1
            beq finic
            sub conteo1
             cmp #2
             blo finic
             jmp ruido
compara1      lda conteo1
             sub siguiente
             cmp #2
             blo finic
ruido         bset 0,flag_ruido
compara       mov conteo1,siguiente
finic         bset 0,flagchequeo
             rti

*****************************************************************************
------------  BASE DE TIEMPO             -----------
*****************************************************************************

GO_ON:   bclr 7,tsc
        brclr 1,puls_enable,go1
        inc contpulsreset
        lda contpulsreset
        cbeqa #!20,finpuls
        brset 0,tx,go1
        rti
finpuls  clr contpulsreset
        bclr 1,puls_enable
        bset ACK1,INTSCR
        bclr IMASK1,INTSCR
        bclr 2,puls_enable
        brset 0,tx,go1
        rti
go1      inc cuenta66mili
        lda cuenta66mili
        cbeqa #1,continua
        cbeqa #2,continua1
        cbeqa #3,continua2
ultimo   lda nrodigito
        cbeqa #4,final
        rti
continua inc nrodigito
        jsr rutina_decide
        clra
        sta tsc0
        brclr 0,flag_ruido,no_ruido
        jsr rutinareset
        lda #$30
```

```
        sta tsc
        clr nrodigito
        clr digito0
        clr fuera_rango
        clr cuenta66mili
        clr newoffset
        clr newoffset2
no_ruido rti
continua1      jsr rutinareset
               rti
continua2 clr cuenta66mili
         clr newoffset
         clr newoffset2
        LDA #$48
        STA TSC0 ;(ENTRADA POR PTD4)
        bra ultimo
final    brset 1,puls_enable,final2
        lda #$30
        sta tsc
final2   clra
        sta tsc0
        clr tx
        clr nrodigito
        bset 6,ptd
        bclr 0,flagchequeo
        brset flag_inic_lcd,lcd_reg,no_inic
        jsr inic_display
        bset flag_inic_lcd,lcd_reg
no_inic  bset do_display,lcd_reg
        rti
siguetono1a     jmp siguetono1
siguetono2a     jmp siguetono2
siguetono3a     jmp siguetono3
siguetono4a     jmp siguetono4
siguetono5a     jmp siguetono5
siguetono6a     jmp siguetono6
siguetono7a     jmp siguetono7
siguetono8a     jmp siguetono8
siguetono9a     jmp siguetono9
siguetonoa1     jmp siguetonoa
siguetonob1     jmp siguetonob
siguetonoc1     jmp siguetonoc
siguetonod1     jmp siguetonod
siguetonoe1     jmp siguetonoe

*****************************************************************************
----------   RUTINA DE DECISIÓN            ----------
*****************************************************************************

rutina_decide   ;USAMOS UNA RUTINA OPTIMIZADA PARA HALLAR EL REGISTRO
```

```
            ; CON EL NRO MAYOR

            lda tono0
-            clrx
            cmp tono1
            blo siguetono1a
sig1          cmp tono2
            blo siguetono2a
sig2          cmp tono3
            blo siguetono3a
sig3          cmp tono4
            blo siguetono4a
sig4          cmp tono5
            blo siguetono5a
sig5          cmp tono6
            blo siguetono6a
sig6          cmp tono7
            blo siguetono7a
sig7          cmp tono8
            blo siguetono8a
sig8          cmp tono9
            blo siguetono9a
sig9          cmp tonoa
            blo siguetonoa1
siga          cmp tonob
            blo siguetonob1
sigb          cmp tonoc
            blo siguetonoc1
sigc          cmp tonod
            blo siguetonod1
sigd          cmp tonoe
            blo siguetonoe1
            cmp fuera_rango
            blo ruido
            bhi decide
            ldx #$f
            jmp decide
ruido         bset 0,flag_ruido
            rts
decide    sta cant_tonos
        LDA   TABLA,X
        sta temp
        lda nrodigito
        cbeqa #$1,led1
        cbeqa #$2,led2
        cbeqa #$3,led3
        cbeqa #$4,led4
        clr nrodigito
led1     lda temp
        cbeqa #$f,ruido
```

```
        ldx cant_tonos
        cpx #$3
        blo ruido
        cbeqa #$45,emer_mode
        mov temp,digito1
        rts
emer_mode mov temp,digito0
         clr nrodigito
         rts
led2     mov temp,digito2
        rts
led3     mov temp,digito3
        rts
led4     mov temp,digito4
        rts

siguetono1      lda tono1
                ldx #$1
                jmp sig1
siguetono2      lda tono2
                ldx #$2
                jmp sig2
siguetono3      lda tono3
                ldx #$3
                jmp sig3
siguetono4      lda tono4
                ldx #$4
                jmp sig4
siguetono5      lda tono5
                ldx #$5
                jmp sig5
siguetono6      lda tono6
                ldx #$6
                jmp sig6
siguetono7      lda tono7
                ldx #$7
                jmp sig7
siguetono8      lda tono8
                ldx #$8
                jmp sig8
siguetono9      lda tono9
                ldx #$9
                jmp sig9
siguetonoa      lda tonoa
                ldx #$a
                jmp siga
siguetonob      lda tonob
                ldx #$b
                jmp sigb
siguetonoc      lda tonoc
```

```
            ldx #$c
            jmp sigc
siguetonod  lda tonod
            ldx #$d
            jmp sigd
siguetonoe  lda tonoe
            ldx #$e
            jmp decide

TABLA   FCB   $30,$31,$32,$33,$34,$35,$36
        FCB   $37,$38,$39,$41,$42,$43,$44,$45,$f

*****************************************************************************
----------   RUTINA DE RESET  DE CONTADORES            ----------
*****************************************************************************

rutinareset   clr tono0
            clr tono1
            clr tono2
            clr tono3
            clr tono4
            clr tono5
            clr tono6
            clr tono7
            clr tono8
            clr tono9
            clr tonoa
            clr tonob
            clr tonoc
            clr tonod
            clr tonoe
            rts

*****************************************************************************
----------   SUBRUTINA DISPLAY DE MENSAJE            ----------
*****************************************************************************

config_char lda nrotxfreez
          beq normal
          cbeqa #1,buff1
          cbeqa #2,buff2
          dec nrotxfreez
normal      jmp signormal
buff1       dec nrotxfreez
          mov #'#',char1
          lda digitof1
          cbeqa #$f,error
          mov digitof1,char2
          mov digitof2,char3
          mov digitof3,char4
```

```
        mov digitof4,char5
        mov #'/',char6
        mov #'S',char7
        lda digitof0
        jmp comun
buff2     dec nrotxfreez
        mov #'#',char1
        lda digitof6
        cbeqa #$f,error
        mov digitof6,char2
        mov digitof7,char3
        mov digitof8,char4
        mov digitof9,char5
        mov #'/',char6
        mov #'S',char7
        lda digitof5
        jmp comun
signormal   mov #'#',char1
        lda digito2
        cbeqa #$f,error
        lda digito3
        cbeqa #$f,error
        lda digito4
        cbeqa #$f,error
        mov digito1,char2
        mov digito2,char3
        mov digito3,char4
        mov digito4,char5
        mov #'/',char6
        mov #'S',char7
        lda digito0
comun     beq sig_normal_mode
        cbeqa #$45,emerg_mode
error     mov #'E',char2
        mov #'R',char3
        mov #'R',char4
        mov #'O',char5
        mov #'R',char6
        mov #' ',char7
        mov #' ',char8
        bra sig_config_char
emerg_mode  mov #'E',char8
        clr digito0
        bset 4,puls_enable
        bra sig_config_char
sig_normal_mode         mov #'N',char8
sig_config_char         brset tx_previa,lcd_reg,clear_flag
                bset tx_previa,lcd_reg
                bclr line,estado
                rts
```

```
clear_flag  bclr tx_previa,lcd_reg
          bset line,estado
          rts

*****************************************************************************
------------   RESET DE DISPLAY              -----------
*****************************************************************************

rutina:
     bset IMASK1,INTSCR
     bset 0,puls_enable
     bset 1,puls_enable
     rti
*-------------------------------------------------------------------
* INICIALIZACION DEL DISPLAY LCD
*-------------------------------------------------------------------
;SUPONEMOS UN RETARDO DE 40 MSEG PARA INICIALIZAR EL DISPLAY
;QUE CONSEGUIMOS DADO QUE EL ENCENDIDO DEL DISPLAY ES INSTANTANEO
;Y LA INICIALIZACION ES DESPUES DE RECIBIR TODOS LOS CODIGOS (200MSEG)
;REALIZARLO EN BASE AL PROGRAMA PCPAL
;EL PROGRAMA NO ESTA OPTIMIZADO EN CUANTO A LAS CONEXIONES
;DADO QUE NO SE HA SIMPLIFICADO LA EQUIVALENCIA DE DATA0/7 A PTB0/7

inic_display    mov #$00,eeadr
              mov #$40,eeadrh
               bclr 2,ptd;ptd2 está conectado a R/W
               bclr 3,ptd ;ptd3 está conectado a RS
               bset 5,ptd
              bset 2,ddrd
              bset 3,ddrd
              bset 5,ddrd
              lda #$30;escribo 00111000 en el display LCD
              sta ptb ;tener en cuenta que DATA7/0 corresponden
              lda #$ff;
              sta ddrb
              bclr 5,ptd
              jsr espera5mili ;espero más de 4,1 mseg
              bset 5,ptd
              bclr 5,ptd ; se activa por flanco descendente
              jsr espera_decimamili ;espero una decima de miliseg.
              lda #$30;escribo 00111000 en el display LCD
              sta ptb
              bclr 2,ptd;ptd2 está conectado a R/W
              bclr 3,ptd ;ptd3 está conectado a RS
              bset 5,ptd ;ptb7 se conecta a E
              bclr 5,ptd;que se activa por flanco descendente
              jsr espera_100micro
              lda #$38;escribo 00111000 en el display LCD FUNCTION SET
              jsr command_write
              lda #$08;escribo 00001000 en el display LCD   DISPLAY OFF
```

```
            jsr command_write
            lda #$01;escribo 00000001 en el display LCD CLEAR DISPLAY              jsr
command_write
            jsr command_write
            lda #$06;escribo 00000110 en el display LCD  ENTRY MODE SET
            jsr command_write

            ;FIN INICIALIZACIÓN DISPLAY

*-------------------------------------------------------------------
*   ENCENDIDO DEL DISPLAY
*-------------------------------------------------------------------
            lda #$0e;escribo 00001110 en el display LCD con cursor titilando
            jsr command_write
           rts
*-------------------------------------------------------------------
*   MENSAJE EN EL DISPLAY
*-------------------------------------------------------------------
          ;Dirección de memoria dada por EEADRH:EEADR
          ;ESTADO,LINE ------ LINEA 1 o 2

message      brclr line,estado,line0
          lda #$40
          bra line1
line0       clra
line1       ora #$80
          jsr command_write
          mov #$04,lcdcont
new_char     bset wmessage,lcd_reg
          bset read,lcd_reg
          nop
          nop
          lda lcdcont
          cbeqa #4,dos_primeros
          cbeqa #3,dos_segundos
          cbeqa #2,dos_terceros
          cbeqa #1,dos_cuartos
dos_primeros  mov char1,lcdtemp
          mov char2,lcdtemp2
          jmp sigue_mens
dos_segundos  mov char3,lcdtemp
          mov char4,lcdtemp2
          jmp sigue_mens
dos_terceros  mov char5,lcdtemp
          mov char6,lcdtemp2
          jmp sigue_mens
dos_cuartos   mov char7,lcdtemp
          mov char8,lcdtemp2
;ENVIO DE LOS DOS CARACTERES
sigue_mens    lda lcdtemp
```

```
            jsr data_write
            lda lcdtemp2
            jsr data_write
;INCREMENTO LA POSICION DE MEMORIA
          brset line,estado,indic_2
          inc eeadr
          dbnz lcdcont,new_char
          rts
indic_2     inc eeadrh
          dbnz lcdcont,new_char
          rts


espera5mili           mov #5,cuenta5mili
siguedec3_5mili       ldx #4  ;base de tiempo de 1mseg
siguedec2_5mili       lda #!171
siguedec_5mili        dbnza siguedec_5mili
                  dbnzx siguedec2_5mili
                  dbnz cuenta5mili,siguedec3_5mili
                  rts
mem_reset             lda #$ff
                  sta ptb
                  rts

espera_decimamili     lda #!82
siguedec              dbnza siguedec
                  rts
espera_100micro       lda #!70
siguedec1             dbnza siguedec1
                  rts
command_write         bclr 3,ptd ;ptd3 está conectado a RS
                  bra write
data_write            bset 3,ptd ;ptd3 está conectado a RS
write               bclr 2,ptd ;ptd2 está conectado a R/W
                  sta ptb
                  bset 5,ptd ;ptd5 se conecta a E
                  bclr 5,ptd ;que se activa por flanco descendente
                  clra
                  sta ddrb;   PONEMOS COMO ENTRADAS A DATA0/DATA7
                  bclr 3,ptd ;ptd3 está conectado a RS
                  bset 2,ptd ;ptd2 está conectado a R/W
                  bset 5,ptd ;ptd5 se conecta a E
bucle               brset 7,ptb,bucle ;equivale a DATA7
                  nop
bucle2              brset 7,ptb,bucle2 ;equivale a DATA7
;bucle3             brset 7,ptb,bucle3 ;equivale a DATA7
                  bclr 5,ptd ;ptd5 se conecta a E
                  lda #$ff
                  sta ddrb
                  rts
```

```
.org reset_vec
   dw start
.org ic_vec
   dw ic_int
.org toverflow
   dw GO_ON
.org irq_int
   dw rutina
```

BIBLIOGRAFÍA

- Apuntes de Cátedra Sistemas Digitales
- Apuntes de Cátedra Electrónica Aplicada II
- Apuntes de Cátedra Mediciones Electrónicas
- Manual Microcontroladores Familia HC08 – MC68HC908JK1/JK3/JL3
- Hojas de datos varias (LM339, LM358, CD4050 –CD4049, etc.)
- Páginas Web

http://www.tml.hut.fi/Studies/T-111.550/2003/Presentations/ITU-ETSI.ppt.

http://www.Voltage Comparator.htm

http://www.Elektronica / Wintek - Specifications 6 for WM-C0802M LCD Module..htm

http://www.baxani120.htm